Universal Serial Bus 3.0 Specification

(including errata and ECNs through May 1, 2011)

Hewlett-Packard Company

Intel Corporation

Microsoft Corporation

NEC Corporation

ST-Ericsson

Texas Instruments

Revision 1.0

June 6, 2011

Revision History

Revision	Comments	Issue Date
1.0	Initial release.	November 12, 2008
	Incorporated errata and ECNs.	June 6, 2011

Please send comments to techsup@usb.org

For industry information, refer to the USB Implementers Forum web page at http://www.usb.org

Acknowledgement of USB 3.0 Technical Contribution

Dedication

Dedicated to the memory of Brad Hosler, the impact of whose accomplishments made the Universal Serial Bus one of the most successful technology innovations of the Personal Computer era.

The authors of this specification would like to recognize the following people who participated in the USB 3.0 Bus Specification technical workgroups. We would also like to acknowledge the many others throughout the industry who provided feedback and contributed to the development of this specification.

Promoter Company Employees

Alan Berkema	Hewlett-Packard Company
Walter Fry	Hewlett-Packard Company
Anthony Hudson	Hewlett-Packard Company
David Roderick	Hewlett-Packard Company
Kok Hong Chan	Intel Corporation
Huimin Chen	Intel Corporation
Bob Dunstan	Intel Corporation
Dan Froelich	Intel Corporation
Howard Heck	Intel Corporation
Brad Hosler	Intel Corporation
John Howard	Intel Corporation
Rahman Ismail	Intel Corporation
John Keys	Intel Corporation
Yun Ling	Intel Corporation
Andy Martwick	Intel Corporation
Steve McGowan	Intel Corporation
Ramin Neshati	Intel Corporation
Duane Quiet	Intel Corporation
Jeff Ravencraft	Intel Corporation
Brad Saunders	Intel Corporation
Joe Schaefer	Intel Corporation
Sarah Sharp	Intel Corporation
Micah Sheller	Intel Corporation
Gary Solomon	Intel Corporation
Karthi Vadivelu	Intel Corporation
Clint Walker	Intel Corporation
Jim Walsh	Intel Corporation
Randy Aull	Microsoft Corporation
Fred Bhesania	Microsoft Corporation
Martin Borve	Microsoft Corporation
Jim Bovee	Microsoft Corporation
Stephen Cooper	Microsoft Corporation
Lars Giusti	Microsoft Corporation
Robbie Harris	Microsoft Corporation

Allen Marshall	Microsoft Corporation
Kiran Muthabatulla	Microsoft Corporation
Tomas Perez-Rodriguez	Microsoft Corporation
Mukund Sankaranarayan	Microsoft Corporation
Nathan Sherman	Microsoft Corporation
Glen Slick	Microsoft Corporation
David Wooten	Microsoft Corporation
Rob Young	Microsoft Corporation
Nobuo Furuya	NEC Corporation
Hiroshi Kariya	NEC Corporation
Masami Katagiri	NEC Corporation
Yuichi Mizoguchi	NEC Corporation
Kats Nakazawa	NEC Corporation
Nobuyuki Mizukoshi	NEC Corporation
Yutaka Noguchi	NEC Corporation
Hajime Nozaki	NEC Corporation
Kenji Oguma	NEC Corporation
Satoshi Ohtani	NEC Corporation
Takanori Saeki	NEC Corporation
Eiji Sakai	NEC Corporation
Hiro Sakamoto	NEC Corporation
Hajime Sakuma	NEC Corporation
Makoto Sato	NEC Corporation
Hock Seow	NEC Corporation
"Peter" Chu Tin Teng	NEC Corporation
Yoshiyuki Tomoda	NEC Corporation
Satomi Yamauchi	NEC Corporation
Yoshiyuki Yamada	NEC Corporation
Susumu Yasuda	NEC Corporation
Alan Chang	ST-NXP Wireless
Wing Yan Chung	ST-NXP Wireless
Socol Constantin	ST-NXP Wireless
Knud Holtvoeth	NXP Semiconductors, B.V.
Linus Kerk	ST-NXP Wireless
Martin Klein	NXP Semiconductors, B.V.
Geert Knapen	NXP Semiconductors, B.V.
Chee Ee Lee	ST-NXP Wireless
Christian Paquet	NXP Semiconductors, B.V.
Veerappan Rajaram	ST-NXP Wireless
Shaun Reemeyer	ST-NXP Wireless
Dave Sroka	ST-NXP Wireless
Chee-Yen TEE	ST-NXP Wireless
Jerome Tjia	ST-NXP Wireless
Bart Vertenten	NXP Semiconductors, B.V.
Hock Meng Yeo	ST-NXP Wireless
Olivier Alavoine	Texas Instruments.
David Arciniega	Texas Instruments
Richard Baker	Texas Instruments
Sujoy Chakravarty	Texas Instruments
T. Y. Chan	Texas Instruments

Romit Dasgupta	Texas Instruments.
Alex Davidson	Texas Instruments
Eric Desmarchelier	Texas Instruments
Christophe Gautier	Texas Instruments
Dan Harmon	Texas Instruments
Will Harris	Texas Instruments
Richard Hubbard	Texas Instruments
Ivo Huber	Texas Instruments
Scott Kim	Texas Instruments
Grant Ley	Texas Instruments
Karl Muth	Texas Instruments
Lee Myers	Texas Instruments
Julie Nirchi	Texas Instruments
Wes Ray	Texas Instruments
Matthew Rowley	Texas Instruments
Bill Sherry	Texas Instruments
Mitsuru Shimada	Texas Instruments
James Skidmore	Texas Instruments
Yoram Solomon	Texas Instruments.
Sue Vining	Texas Instruments
Jin-sheng Wang	Texas Instruments
Roy Wojciechowski	Texas Instruments

Contributor Company Employees

Glen Chandler	Acon
John Chen	Acon
Roger Hou	Acon
Charles Wang	Acon
Norman Wu	Acon
Steven Yang	Acon
George Yee	Acon
George Olear	Contech Research
Sophia Liu	Electronics Testing Center, Taiwan (ETC)
William Northey	FCI
Tom Sultzer	FCI
Garry Biddle	Foxconn
Kuan-Yu Chen	Foxconn
Jason Chou	Foxconn
Gustavo Duenas	Foxconn
Bob Hall	Foxconn
Jiayong He	Foxconn
Jim Koser	Foxconn
Joe Ortega	Foxconn
Ash Raheja	Foxconn
James Sabo	Foxconn
Pei Tsao	Foxconn
Kevin Walker	Foxconn
Tsuneki Watanabe	Foxconn
Chong Yi	Foxconn
Taro Hishinuma	Hirose Electric
Kaz Ichikawa	Hirose Electric
Ryozo Koyama	Hirose Electric
Karl Kwiat	Hirose Electric
Tadashi Sakaizawa	Hirose Electric
Shinya Tono	Hirose Electric

Eiji Wakatsuki	Hirose Electric
Takashi Ehara	Japan Aviation Electronics Industry Ltd. (JAE)
Ron Muir	Japan Aviation Electronics Industry Ltd. (JAE)
Kazuhiro Saito	Japan Aviation Electronics Industry Ltd. (JAE)
Hitoshi Kawamura	Mitsumi
Takashi Kawasaki	Mitsumi
Atsushi Nishio	Mitsumi
Yasuhiko Shinohara	Mitsumi
Tom Lu	Molex Inc.
Edmund Poh	Molex Inc.
Scott Sommers	Molex Inc.
Jason Squire	Molex Inc.
Dat Ba Nguyen	NTS/National Technical System
Jan Fahllund	Nokia
Richard Petrie	Nokia
Panu Ylihaavisto	Nokia
Martin Furuhjelm	Seagate Technology LLC
Julian Gorfajn	Seagate Technology LLC
Marc Hildebrant	Seagate Technology LLC
Tony Priborsky	Seagate Technology LLC
Harold To	Seagate Technology LLC
Robert Lefferts	Synopsys, Inc.
Saleem Mohammad	Synopsys, Inc.
Matthew Myers	Synopsys, Inc.
Daniel Weinlader	Synopsys, Inc.
Mike Engbretson	Tektronix, Inc.
Thomas Grzysiewicz	Tyco Electronics
Masaaki Iwasaki	Tyco Electronics
Kazukiyo Osada	Tyco Electronics
Hiroshi Shirai	Tyco Electronics
Scott Shuey	Tyco Electronics
Masaru Ueno	Tyco Electronics

Contents

Figures

Tables

1 Introduction

1.1 Motivation

The original motivation for the Universal Serial Bus (USB) came from several considerations, two of the most important being:

- **Ease-of-use**

  The lack of flexibility in reconfiguring the PC had been acknowledged as the Achilles' heel to its further deployment. The combination of user-friendly graphical interfaces and the hardware and software mechanisms associated with new-generation bus architectures have made computers less confrontational and easier to reconfigure. However, from the end user's point of view, the PC's I/O interfaces, such as serial/parallel ports, keyboard/mouse/joystick interfaces, etc., did not have the attributes of plug-and-play.

- **Port Expansion**

  The addition of external peripherals continued to be constrained by port availability. The lack of a bidirectional, low-cost, low-to-mid speed peripheral bus held back the creative proliferation of peripherals such as storage devices, answering machines, scanners, PDA's, keyboards, and mice. Existing interconnects were optimized for one or two point products. As each new function or capability was added to the PC, a new interface had been defined to address this need.

Initially, USB provided two speeds (12 Mb/s and 1.5 Mb/s) that peripherals could use. As PCs became increasingly powerful and able to process larger amounts of data, users needed to get more and more data into and out of their PCs. This led to the definition of the USB 2.0 specification in 2000 to provide a third transfer rate of 480 Mb/s while retaining backward compatibility. In 2005, with wireless technologies becoming more and more capable, Wireless USB was introduced to provide a new cable free capability to USB.

USB is the most successful PC peripheral interconnect ever defined and it has migrated heavily into the CE and Mobile segments. In 2006 alone over 2 billion USB devices were shipped and there are over 6 billion USB products in the installed base today. End users "know" what USB is. Product developers understand the infrastructure and interfaces necessary to build a successful product.

USB has gone beyond just being a way to connect peripherals to PCs. Printers use USB to interface directly to cameras. PDAs use USB connected keyboards and mice. The USB On-The-Go definition provides a way for two dual role capable devices to be connected and negotiate which one will operate as the "host." USB, as a protocol, is also being picked up and used in many nontraditional applications such as industrial automation.

Now, as technology innovation marches forward, new kinds of devices, media formats, and large inexpensive storage are converging. They require significantly more bus bandwidth to maintain the interactive experience users have come to expect. HD Camcorders will have tens of gigabytes of storage that the user will want to move to their PC for editing, viewing, and archiving. Furthermore existing devices like still image cameras continue to evolve and are increasing their storage capacity to hold even more uncompressed images. Downloading hundreds or even thousands of 10 MB, or larger, raw images from a digital camera will be a time consuming process unless the

transfer rate is increased. In addition, user applications demand a higher performance connection between the PC and these increasingly sophisticated peripherals. USB 3.0 addresses this need by adding an even higher transfer rate to match these new usages and devices.

Thus, USB (wired or wireless) continues to be the answer to connectivity for PC, Consumer Electronics, and Mobile architectures. It is a fast, bidirectional, low-cost, dynamically attachable interface that is consistent with the requirements of the PC platforms of today and tomorrow.

1.2 Objective of the Specification

This document defines the next generation USB industry-standard, USB 3.0. The specification describes the protocol definition, types of transactions, bus management, and the programming interface required to design and build systems and peripherals that are compliant with this specification.

USB 3.0's goal remains to enable devices from different vendors to interoperate in an open architecture, while maintaining and leveraging the existing USB infrastructure (device drivers, software interfaces, etc.). The specification is intended as an enhancement to the PC architecture, spanning portable, business desktop, and home environments, as well as simple device-to-device communications. It is intended that the specification allow system OEMs and peripheral developers adequate room for product versatility and market differentiation without the burden of carrying obsolete interfaces or losing compatibility.

1.3 Scope of the Document

The specification is primarily targeted at peripheral developers and platform/adapter developers, but provides valuable information for platform operating system/ BIOS/ device driver, adapter IHVs/ISVs, and system OEMs. This specification can be used for developing new products and associated software.

Product developers using this specification are expected to know and understand the USB 2.0 Specification. Specifically, USB 3.0 devices must implement device framework commands and descriptors as defined in the *USB 2.0 Specification*.

1.4 USB Product Compliance

Adopters of the USB 3.0 specification have signed the USB 3.0 Adopters Agreement, which provides them access to a reasonable and nondiscriminatory (RANDZ) license from the Promoters and other Adopters to certain intellectual property contained in products that are compliant with the USB 3.0 specification. Adopters can demonstrate compliance with the specification through the testing program as defined by the USB Implementers Forum. Products that demonstrate compliance with the specification will be granted certain rights to use the USB Implementers Forum logos as defined in the logo license.

1.5 Document Organization

Chapters 1 through 4 provide an overview for all readers, while Chapters 5 through 11 contain detailed technical information defining USB 3.0.

Readers should contact operating system vendors for operating system bindings specific to USB 3.0.

1.6 Design Goals

USB 3.0 is the next evolutionary step for wired USB. The goal is that end users view it as the same as USB 2.0, just faster. Several key design areas to meet this goal are listed below:

- Preserve the USB model of smart host and simple device.
- Leverage the existing USB infrastructure. There are a vast number of USB products in use today. A large part of their success can be traced to the existence of stable software interfaces, easily developed software device drivers, and a number of generic standard device class drivers (HID, mass storage, audio, etc.). SuperSpeed USB devices are designed to keep this software infrastructure intact so that developers of peripherals can continue to use the same interfaces and leverage all of their existing development work.
- Significantly improve power management. Reduce the active power when sending data and reduce idle power by providing a richer set of power management mechanisms to allow devices to drive the bus into lower power states.
- Ease of use has always been and remains a key design goal for all varieties of USB.
- Preserve the investment. There are a large number of PCs in use that support only USB 2.0. There are a larger number of USB 2.0 peripherals in use. Retaining backward compatibility at the Type-A connector to allow SuperSpeed devices to be used, albeit at a lower speed, with USB 2.0 PCs and allow high speed devices with their existing cables to be connected to the USB 3.0 SuperSpeed Type-A connectors.

1.7 Related Documents

Universal Serial Bus Specification, Revision 2.0

USB On-the-Go Supplement to the USB 2.0 Specification, Revision 1.3

USB On-the-Go and Embedded Host Supplement to the USB 3.0 Specification, Revision 1.0

Universal Serial Bus Micro-USB Cables and Connectors Specification, Revision 1.01

EIA-364-1000.01: Environmental Test Methodology for Assessing the Performance of Electrical Connectors and Sockets Used in Business Office Applications

USB 3.0 Connectors and Cable Assemblies Compliance Document

USB SuperSpeed Electrical Test Methodology white paper

USB 3.0 Jitter Budgeting white paper

INCITS TR-35-2004, *INCITS Technical Report for Information Technology – Fibre Channel – Methodologies for Jitter and Signal Quality Specification (FC-MJSQ)*

2 Terms and Abbreviations

This chapter lists and defines terms and abbreviations used throughout this specification.

Term/Abbreviation	Definition
ACK	Handshake packet indicating a positive acknowledgment.
ACK Tx Header Sequence Number	The expected header sequence number in the link control word to be acknowledged.
active device	A device that is powered and is not in the Suspend state.
asynchronous data	Data transferred at irregular intervals with relaxed latency requirements.
attached	A downstream device is attached to an upstream device when there is a physical connection between the two.
AWG#	The measurement of a wire's cross section, as defined by the American Wire Gauge standard.
bandwidth	The amount of data transmitted per unit of time, typically bits per second (bps) or bytes per second (Bps).
big endian	A method of storing data that places the most significant byte of multiple-byte values at a lower storage address. For example, a 16-bit integer stored in big endian format places the least significant byte at the higher address and the most significant byte at the lower address. See also little endian.
bit	A unit of information used by digital computers. Represents the smallest piece of addressable memory within a computer. A bit expresses the choice between two possibilities and is typically represented by a logical one (1) or zero (0).
bps	Transmission rate expressed in bits per second.
Bps	Transmission rate expressed in bytes per second.
buffer	Storage used to compensate for a difference in data rates or time of occurrence of events, when transmitting data from one device to another.
bulk transfer	One of the four USB transfer types. Bulk transfers are non-periodic, large bursty communication typically used for a transfer that can use any available bandwidth and can also be delayed until bandwidth is available. See also transfer type.
bus enumeration	Detecting, identifying, and configuring USB devices.
bus interval	A 125 µs period that establishes the integral boundary of service intervals.
byte	A data element that is 8 bits in size.
cable	Raw cable with no plugs attached.
cable assembly	Cable attached with plugs.
captive cable	Cable assembly that has a Type-A plug on one end and that is either permanently attached or has a vendor specific connector on the other end.
capabilities	Those attributes of a USB device that are administrated by the host.
CDR	Circuit that performs the Clock and Data Recovery function.
characteristics	Those qualities of a USB device that are unchangeable; for example, the device class is a device characteristic.
client	Software resident on the host that interacts with the USB system software to arrange data transfer between a function and the host. The client is often the data provider and consumer for transferred data.
component	A physical chip or circuit that contains a port.

Term/Abbreviation	Definition
configuring software	Software resident on the host that is responsible for configuring a USB device.
control endpoint	A pair of device endpoints with the same endpoint number that are used by a control pipe. Control endpoints transfer data in both directions and, therefore, use both endpoint directions of a device address and endpoint number combination. Thus, each control endpoint consumes two endpoint addresses.
control pipe	Same as a message pipe.
connected	A downstream device is connected to an upstream device when it is attached to the upstream device, and when the downstream device has asserted Rx terminations for SuperSpeed signaling or has asserted the D+ or D- data line in order to enter low-speed, full-speed, or high-speed signaling.
control transfer	One of the four USB transfer types. Control transfers support configuration/command/status type communications between client and function. See also transfer type.
Controlling Hub	A controlling hub is any hub whose upstream link is not in U3.
CRC	CRC-5, CRC-16, CRC-32. See Cyclic Redundancy Check.
Cyclic Redundancy Check (CRC)	A check performed on data to see if an error has occurred in transmitting, reading, or writing the data. The result of a CRC is typically stored or transmitted with the checked data. The stored or transmitted result is compared to a CRC calculated from the data to determine if an error has occurred.
D codes	The data type codes used in 8b/10b encoding.
D+ and D-	Differential pair defined in the USB 2.0 specification.
default address	An address defined by the USB Specification and used by a USB device when it is first powered or reset. The default address is 00H.
default pipe	The message pipe created by the USB system software to pass control and status information between the host and a USB device's endpoint zero.
descrambling	Restoring the pseudo-random 8-bit character to the original state. See scrambling.
detached	A downstream device is detached from an upstream device when the physical cable between the two is removed.
device	A logical or physical entity that performs one or more functions. The actual entity described depends on the context of the reference. At the lowest level, device may refer to a single hardware component, as in a memory device. At a higher level, it may refer to a collection of hardware components that perform a particular function, such as a USB interface device. At an even higher level, device may refer to the function performed by an entity attached to the USB. Devices may be physical, electrical, addressable, and logical. When used as a non-specific reference, a USB device is either a hub or a peripheral device.
device address	A 7-bit value representing the address of a device on the USB. The device address is the default address (00H) when the USB device is first powered or the device is reset. Devices are assigned a unique device address by the USB system software.
device endpoint	A uniquely addressable portion of a USB device that is the source or sink of information in a communication flow between the host and device. See also endpoint address.
device software	Software that is responsible for using a USB device. This software may or may not also be responsible for configuring the device for use.
disconnected (unconnected)	A downstream device is disconnected from an upstream device when it is attached to the upstream device, and when the downstream device has not asserted Rx terminations for SuperSpeed signaling or has not asserted either the D+ or D- data line in order to enter low-speed, full-speed, or high-speed signaling.

Term/Abbreviation	Definition
downstream	The direction of data flow from the host or away from the host. A downstream port is the port on a hub electrically farthest from the host that generates downstream data traffic from the hub. Downstream ports receive upstream data traffic.
downstream facing port	See downstream port.
downstream port	The port on a host or a hub to which a device is connected. For example, a system's root ports are downstream ports.
DP	Data Packet which consists of a Data Packet Header followed by a Data Packet Payload.
DPH	Data Packet Header. Contains the data packet's address, route string, length, and other information about the packet.
DPP	Data Packet Payload. Contains the data packet's data and a 32-bit CRC.
DPPABORT	Frame ordered set used to abort a data packet payload.
DPPEND	Frame ordered set used to denote the end of a data packet payload.
DPPSTART	Frame ordered set used to denote the start of a data packet payload.
driver	When referring to hardware, an I/O pad that drives an external load. When referring to software, a program responsible for interfacing to a hardware device, that is, a device driver.
DSPORT	Notation indicating the state machine of a downstream facing port of a hub. Refer to Section 10.3.
dual simplex	Two data paths that independently carry traffic in each direction.
DWORD	Double word. A data element that is two words (i.e., 4 bytes or 32 bits) in size.
dynamic insertion and removal	The ability to attach and remove devices while the host is in operation.
endpoint	See device endpoint.
endpoint address	The combination of an endpoint number and an endpoint direction on a USB device. Each endpoint address supports data transfer in one direction.
endpoint direction	The direction of data transfer on the USB. The direction can be either IN or OUT. IN refers to transfers to the host; OUT refers to transfers from the host.
endpoint number	A four-bit value between 0H and FH, inclusive, associated with an endpoint on a USB device.
external port	See port.
frame number	The bus interval counter value within the ITP divided by 8 (integer division).
full-duplex	Computer data transmission occurring in both directions simultaneously.
full-speed	USB operation at 12 Mbps. See also low-speed and high-speed.
function	A set of one or more related interfaces on a USB device that exposes a capability to a software client.
Gbps	Transmission rate expressed in gigabits per second (1,000,000,000 bits per second).
handshake packet	A packet that acknowledges or rejects a specific condition. For examples, see ACK, NRDY, or ERDY.
header	Packet header. For example, DPH, LMP, and TP are all headers.
Header Sequence Number Advertisement	The exchange of the ACK Tx Header Sequence Numbers between the link partners upon entry to U0.
high-speed	USB operation at 480 Mbps. See also low-speed and full-speed.
host	The host computer system where the USB host controller is installed. This includes the host hardware platform (CPU, bus, etc.) and the operating system in use.
host controller	The interface provided to the system to support devices on the USB.

Term/Abbreviation	Definition
Hot Reset	Reset mechanism using TS1/TS2 ordered sets.
HPSTART	Frame ordered set used to denote the start of a header packet.
hub	A USB device that provides additional connections to the USB.
hub tier	One plus the number of USB links in a communication path between the host and a peripheral device.
ID pin	Denotes the pin on the USB 3.0 Micro connector family that is used to differentiate a USB 3.0 Micro-A plug from a USB 3.0 Micro-B plug.
Inband Reset	Mechanism that relies on SuperSpeed and/or LFPS signaling to propagate the reset across the link.
informative	Information given for illustrative purposes only and contains no requirements. See normative.
interrupt transfer	One of the four USB transfer types. Interrupt transfers have a bounded latency and are typically used to handle service needs. See also transfer type.
isochronous data	A stream of data whose timing is implied by its delivery rate.
isochronous device	An entity with isochronous endpoints, as defined in the USB Specification, that sources or sinks sampled analog streams or synchronous data streams.
isochronous sink endpoint	An endpoint that is capable of consuming an isochronous data stream that is sent by the host.
isochronous source endpoint	An endpoint that is capable of producing an isochronous data stream and sending it to the host.
isochronous transfer	One of the four USB transfer types. Isochronous transfers are used when working with isochronous data. Isochronous transfers provide periodic, continuous communication between host and device. See also transfer type.
ITP	Isochronous Timestamp Packet, sent periodically by a host to inform devices on the USB of the current bus time.
jitter	A tendency toward lack of synchronization caused by mechanical or electrical changes. More specifically, the phase shift of digital pulses over a transmission medium.
KB	Kilobyte or 1,024 bytes.
K codes	The control type codes used in 8b/10b encoding. SHP – start header packet SDP – start data packet END – end header or data packet EDB – end of nullified (bad) packet SLC – start link command COM – comma SKP – skip EPF – end packet framing
LCSTART	Frame ordered set used to denote the start of a link command.
LFPS	Low frequency periodic signal. Used to communicate information across a link without using SuperSpeed signaling.
LFSR	Linear feedback shift register. Used to create pseudo-random characters for scrambling.
LI	Logical Idle
link command	An eight-symbol sequence used for link-level flow control, retries, power management, and device removal.

Term/Abbreviation	Definition
Link Control Word	Two bytes with 11 bits to define the link level flow control and a 5-bit CRC5 to ensure data integrity.
little endian	Method of storing data that places the least significant byte of multiple-byte values at lower storage addresses. For example, a 16-bit integer stored in little endian format places the least significant byte at the lower address and the most significant byte at the next address. See also big endian.
LMP	Link Management Packet. A type of header packet used to communicate information between a pair of links.
Local Rx Header Buffer Credit	The availability of a single free Rx Header Buffer of a port itself.
Logical Idle	Period of one or more symbol times when no information (packets or link commands) is being transmitted when link is in U0.
low-speed	USB operation at 1.5 Mbps. See also full-speed and high-speed.
LSb	Least significant bit.
LSB	Least significant byte.
LTSSM	Link Training and Status State Machine.
message pipe	A bi-directional pipe that transfers data using a request/data/status paradigm. The data has an imposed structure that allows requests to be reliably identified and communicated.
MSb	Most significant bit.
MSB	Most significant byte.
normative	Required by the specification. See also informative.
NRDY	Handshake packet indicating a negative acknowledgment.
packet	A bundle of data organized in a group for transmission. Packets typically contain three elements: control information (e.g., source, destination, and length), the data to be transferred and error detection and correction bits.
peripheral	A physical entity that is attached to a USB cable and is currently operating as a “device” as defined in this specification.
peripheral device	A non-hub USB device that provides one or more functions to the host, such as a mass storage device.
persistent	State information (e.g., a descriptor field) that is retained and persistent through entry into and exit from D3.
Phase Locked Loop	A circuit that acts as a phase detector to keep an oscillator in phase with an incoming frequency.
physical device	A device that has a physical implementation; e.g., speakers, microphones, and CD players.
pipe	A logical abstraction representing the association between an endpoint on a device and software on the host. A pipe has several attributes; for example, a pipe may transfer data as streams (stream pipe) or messages (message pipe). See also stream pipe and message pipe.
PLL	See Phase Locked Loop.
plug	Connector attached to the cable, to be mated with the receptacle
port	Point of access to or from a system or circuit. For the USB, the point where a USB device is attached.
PowerOn Reset (POR)	An event to restore a device to its initial state.
PPM	Parts Per Million.
PRBS	Pseudo-Random Bit Stream.

Term/Abbreviation	Definition
protocol	A specific set of rules, procedures, or conventions relating to format and timing of data transmission between two devices.
receptacle	Connector mounted on the host or device, to be mated with the plug.
Remote Rx Header Buffer Credit	The availability of a single free Rx Header Buffer from a link partner.
request	A request made to a USB device contained within the data portion of a SETUP packet.
root hub	A USB hub directly attached to or integrated into the host controller.
root port	The downstream port on a root hub.
Rx Header Buffer Credit Advertisement	The exchange of the Remote Rx Header Buffer Credits between the link partners upon entry to U0.
Rx Header Sequence Number	The expected header sequence number of a header packet received from a link partner.
scrambling	The process of changing an eight-bit character in a pseudo-random way. See descrambling.
SDP	Shielded Differential Pair.
service interval	An integral multiple of bus intervals within which a periodic endpoint must be serviced.
service jitter	The deviation of service delivery from its scheduled delivery time.
SSC	Spread Spectrum Clock.
stage	One part of the sequence composing a control transfer; stages include the Setup stage, the Data stage, and the Status stage.
stream endpoint	A SuperSpeed bulk endpoint whose SuperSpeed Endpoint Companion Descriptor bmAtttributes field declares a MaxStreams value that is greater than 0.
stream pipe	A pipe that transfers data as a stream of samples with no defined USB structure.
SuperSpeed	USB operation at 5 Gbps.
synchronization type	A classification that characterizes an isochronous endpoint's capability to connect to other isochronous endpoints.
termination	Passive components attached at the end of the connections to prevent signals from being reflected or echoed.
timeout	A time interval within which an expected event shall occur.
TP	Transaction Packet. A type of header packet used to communicate information between a device and the host.
training sequences	Ordered sets for initializing bit and symbol alignment and receiver equalization. Examples are TS1, TS2, and TSEQ.
transaction	The delivery of service to an endpoint: • The IN consists of an ACK TP with a response of NRDY TP, DP, or STALL TP. • The OUT consists of a DP with a response of NRDY TP, an ACK TP, or STALL TP.
transfer	One or more bus transactions to move information between a software client and its function.
transfer type	Determines the characteristics of the data flow between a software client and its function. Four standard transfer types are defined: control, interrupt, bulk, and isochronous.
Type-A connector	The standard-A connector defined in this specification.
Tx Header Sequence Number	The header sequence number to be added to a header packet to be transmitted.
upstream	The direction of data flow towards the host. An upstream port is the port on a device electrically closest to the host. Upstream ports receive downstream data traffic.

Term/Abbreviation	Definition
upstream port	A port that a device uses to connect to a host or a hub. The port on all devices is an upstream port.
upstream facing port	See upstream port.
USB 3.0 Standard-A connector	USB 3.0 host connector, supporting SuperSpeed mode.
USB 3.0 Powered-B connector	The standard Type-B device connector, supporting USB 3.0 SuperSpeed mode with additional pins for power delivery from the device.
USB 3.0 Standard-B connector	The standard Type-B device connector, supporting USB 3.0 SuperSpeed mode.
USB 3.0 Micro-A plug	Part of the USB 3.0 Micro connector family for OTG use; it can be plugged into a USB 3.0 Micro-AB receptacle; it differs from the USB 3.0 Micro-B plug only in keying and ID pin connection.
USB 3.0 Micro-AB receptacle	Part of the USB 3.0 Micro connector family; it accepts either a USB 3.0 Micro-B plug or a USB 3.0 Micro-A plug.
USB 3.0 Micro-B connector	USB 3.0 device connector, supporting SuperSpeed mode.
USB 3.0 Micro connector family	All the receptacles and plugs that are used on devices, including the USB 3.0 Micro-B, USB 3.0 Micro-AB, and USB 3.0 Micro-A connectors.
USB 2.0 Standard-A connector	The Type-A connector defined by the USB 2.0 specification.
USB 2.0 Standard-B connector	The standard Type-B connector defined by the USB 2.0 specification.
USB-IF	USB Implementers Forum, Inc. is a nonprofit corporation formed to facilitate the development of USB compliant products and promote the technology.
USDPORT	Notation indicating the state machine of the upstream facing port of a peripheral device. Refer to Section 10.16.
USPORT	Notation indicating the state machine of the upstream facing port of a hub. Refer to Section 10.5.
UTP	Unshielded Twisted Pair.
Warm Reset	Reset mechanism using LFPS.
WORD	A data element that is 2 bytes (16 bits) in size.

3 USB 3.0 Architectural Overview

This chapter presents an overview of Universal Serial Bus 3.0 architecture and key concepts. USB 3.0 is similar to earlier versions of USB in that it is a cable bus supporting data exchange between a host computer and a wide range of simultaneously accessible peripherals. The attached peripherals share bandwidth through a host-scheduled protocol. The bus allows peripherals to be attached, configured, used, and detached while the host and other peripherals are in operation.

USB 3.0 utilizes a dual-bus architecture that provides backward compatibility with USB 2.0. It provides for simultaneous operation of SuperSpeed and non-SuperSpeed (USB 2.0 speeds) information exchanges. This chapter is organized into two focus areas. The first focuses on architecture and concepts related to elements which span the dual buses. The second focuses on SuperSpeed specific architecture and concepts.

Later chapters describe the various components and specific requirements of SuperSpeed USB in greater detail. The reader is expected to have a fundamental understanding of the architectural concepts of USB 2.0. Refer to the *Universal Serial Bus Specification, Revision 2.0* for complete details.

3.1 USB 3.0 System Description

USB 3.0 is a physical SuperSpeed bus combined in parallel with a physical USB 2.0 bus (see Figure 3-1). It has similar architectural components as USB 2.0, namely:

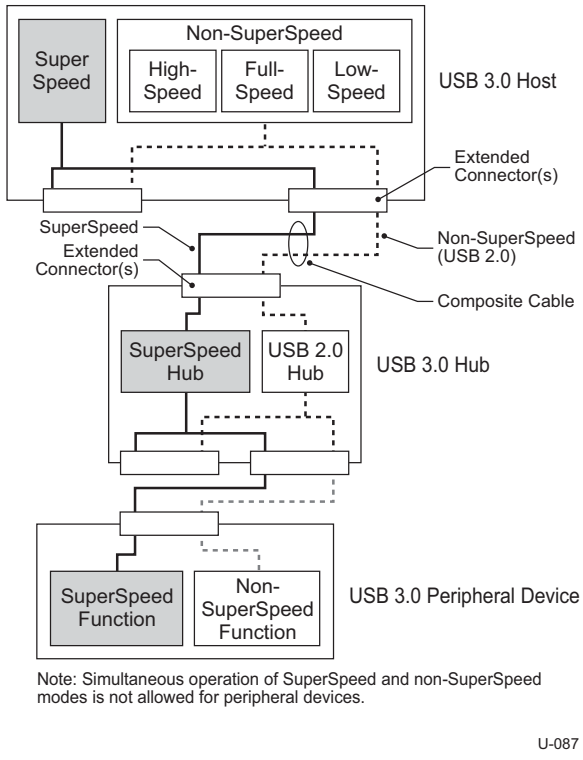


Figure 3-1. USB 3.0 Dual Bus Architecture

- USB 3.0 interconnect
- USB 3.0 devices
- USB 3.0 host

The USB 3.0 interconnect is the manner in which USB 3.0 and USB 2.0 devices connect to and communicate with the USB 3.0 host. The USB 3.0 interconnect inherits core architectural elements from USB 2.0, although several are augmented to accommodate the dual bus architecture.

The baseline structural topology is the same as USB 2.0. It consists of a tiered star topology with a single host at tier 1 and hubs at lower tiers to provide bus connectivity to devices.

The USB 3.0 connection model accommodates backwards and forward compatibility for connecting USB 3.0 or USB 2.0 devices into a USB 3.0 bus. Similarly, USB 3.0 devices can be attached to a USB 2.0 bus. The mechanical and electrical backward/forwards compatibility

for USB 3.0 is accomplished via a composite cable and associated connector assemblies that form the dual-bus architecture. USB 3.0 devices accomplish backward compatibility by including both SuperSpeed and non-SuperSpeed bus interfaces. USB 3.0 hosts also include both SuperSpeed and non-SuperSpeed bus interfaces, which are essentially parallel buses that may be active simultaneously.

The USB 3.0 connection model allows for the discovery and configuration of USB devices at the highest signaling speed supported by the device, the highest signaling speed supported by all hubs between the host and device, and the current host capability and configuration.

USB 3.0 hubs are a specific class of USB device whose purpose is to provide additional connection points to the bus beyond those provided by the host. In this specification, non-hub devices are referred to as peripheral devices in order to differentiate them from hub devices. In addition, in USB 2.0 the term “function” was sometimes used interchangeably with device. In this specification a function is a logical entity within a device, see Figure 3-3.

The architectural implications of SuperSpeed on hosts and devices are described in detail in Section 3.2.

3.1.1 USB 3.0 Physical Interface

The physical interface of USB 3.0 is comprised of USB 2.0 electrical (Chapter 7 of the USB 2.0 specification), mechanical (Chapter 5), and SuperSpeed physical (Chapter 6) specifications for the buses. The SuperSpeed physical layer is described in Section 3.2.1.

3.1.1.1 USB 3.0 Mechanical

The mechanical specifications for USB 3.0 cables and connector assemblies are provided in Chapter 5. All USB devices have an upstream connection. Hosts and hubs (defined below) have one or more downstream connections. Upstream and downstream connectors are not mechanically interchangeable, thus eliminating illegal loopback connections at hubs.

USB 3.0 cables have eight primary conductors: three twisted signal pairs for USB data paths and a power pair. Figure 3-2 illustrates the basic signal arrangement for the USB 3.0 cable. In addition to the twisted signal pair for USB 2.0 data path, two twisted signal pairs are used to provide the SuperSpeed data path, one for the transmit path and one for the receive path.

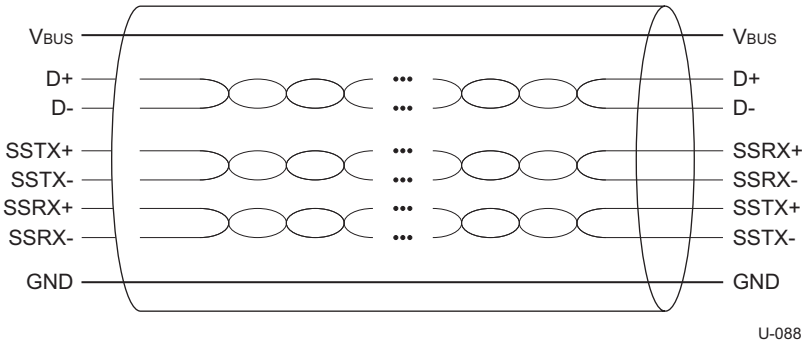


Figure 3-2. USB 3.0 Cable

USB 3.0 receptacles (both upstream and downstream) are backward compatible with USB 2.0 connector plugs. USB 3.0 cables and plugs are not intended to be compatible with USB 2.0 upstream receptacles. As an aid to users, USB 3.0 mandates standard coloring for plastic portions of USB 3.0 plugs and receptacles.

Electrical (insertion loss, return loss, crosstalk, etc.) performance for USB 3.0 is defined with regard to raw cables, mated connectors, and mated cable assemblies, with compliance requirements using industry test specifications established for the latter two categories. Similarly, mechanical (insertion/extraction forces, durability, etc.) and environmental (temperature life, mixed flowing gas, etc.) requirements are defined and compliance established via recognized industry test specifications.

3.1.2 USB 3.0 Power

The specification covers two aspects of power:

- Power distribution over the USB deals with the issues of how USB devices consume power provided by the downstream ports to which they are connected. USB 3.0 power distribution is similar to USB 2.0, with increased supply budgets for devices operating at SuperSpeed.
- Power management deals with how hosts, devices, hubs, and the USB system software interact to provide power efficient operation of the bus. The power management of the USB 2.0 bus portion is unchanged. The use model for power management of the SuperSpeed bus is described in Appendix C.

3.1.3 USB 3.0 System Configuration

USB 3.0 supports USB devices (all speeds) attaching and detaching from the USB 3.0 at any time. Consequently, system software must accommodate dynamic changes in the physical bus topology. The architectural elements for the discovery of attachment and removal of devices on USB 3.0 are identical to those in USB 2.0. There are enhancements provided to manage the specifics of the SuperSpeed bus for configuration and power management.

The independent, dual-bus architecture allows for activation of each of the buses independently and provides for the attachment of USB devices to the highest speed bus available for the device.

3.1.4 USB 3.0 Architecture Summary

USB 3.0 is a dual-bus architecture that incorporates USB 2.0 and a SuperSpeed bus. Table 3-1 summarizes the key architectural differences between SuperSpeed USB and USB 2.0.

Table 3-1. Comparing SuperSpeed to USB 2.0

Characteristic	SuperSpeed USB	USB 2.0
Data Rate	SuperSpeed (5.0 Gbps)	low-speed (1.5 Mbps), full-speed (12 Mbps), and high-speed (480 Mbps)
Data Interface	Dual-simplex, four-wire differential signaling separate from USB 2.0 signaling Simultaneous bi-directional data flows	Half-duplex two-wire differential signaling Unidirectional data flow with negotiated directional bus transitions
Cable signal count	Six: Four for SuperSpeed data path Two for non-SuperSpeed data path	Two: Two for low-speed/full-speed/high-speed data path
Bus transaction protocol	Host directed, asynchronous traffic flow Packet traffic is explicitly routed	Host directed, polled traffic flow Packet traffic is broadcast to all devices.

Characteristic	SuperSpeed USB	USB 2.0
Power management	Multi-level link power management supporting idle, sleep, and suspend states. Link-, Device-, and Function-level power management.	Port-level suspend with two levels of entry/exit latency Device-level power management
Bus power	Same as for USB 2.0 with a 50% increase for unconfigured power and an 80% increase for configured power	Support for low/high bus-powered devices with lower power limits for un-configured and suspended devices
Port State	Port hardware detects connect events and brings the port into operational state ready for SuperSpeed data communication.	Port hardware detects connect events. System software uses port commands to transition the port into an enabled state (i.e., can do USB data communication flows).
Data transfer types	USB 2.0 types with SuperSpeed constraints. Bulk has streams capability (refer to Section 3.2.8)	Four data transfer types: control, bulk, Interrupt, and Isochronous

3.2 SuperSpeed Architecture

The SuperSpeed bus is a layered communications architecture that is comprised of the following elements:

- **SuperSpeed Interconnect.** The SuperSpeed interconnect is the manner in which devices are connected to and communicate with the host over the SuperSpeed bus. This includes the topology of devices connected to the bus, the communications layers, the relationships between them and how they interact to accomplish information exchanges between the host and devices.
- **Devices.** SuperSpeed devices are sources or sinks of information exchanges. They implement the required device-end, SuperSpeed communications layers to accomplish information exchanges between a driver on the host and a logical function on the device.
- **Host.** A SuperSpeed host is a source or sink of information. It implements the required host-end, SuperSpeed communications layers to accomplish information exchanges over the bus. It owns the SuperSpeed data activity schedule and management of the SuperSpeed bus and all devices connected to it.

Figure 3-3 illustrates a reference diagram of the SuperSpeed interconnect represented as communications layers through a topology of host, zero to five levels of hubs, and devices.

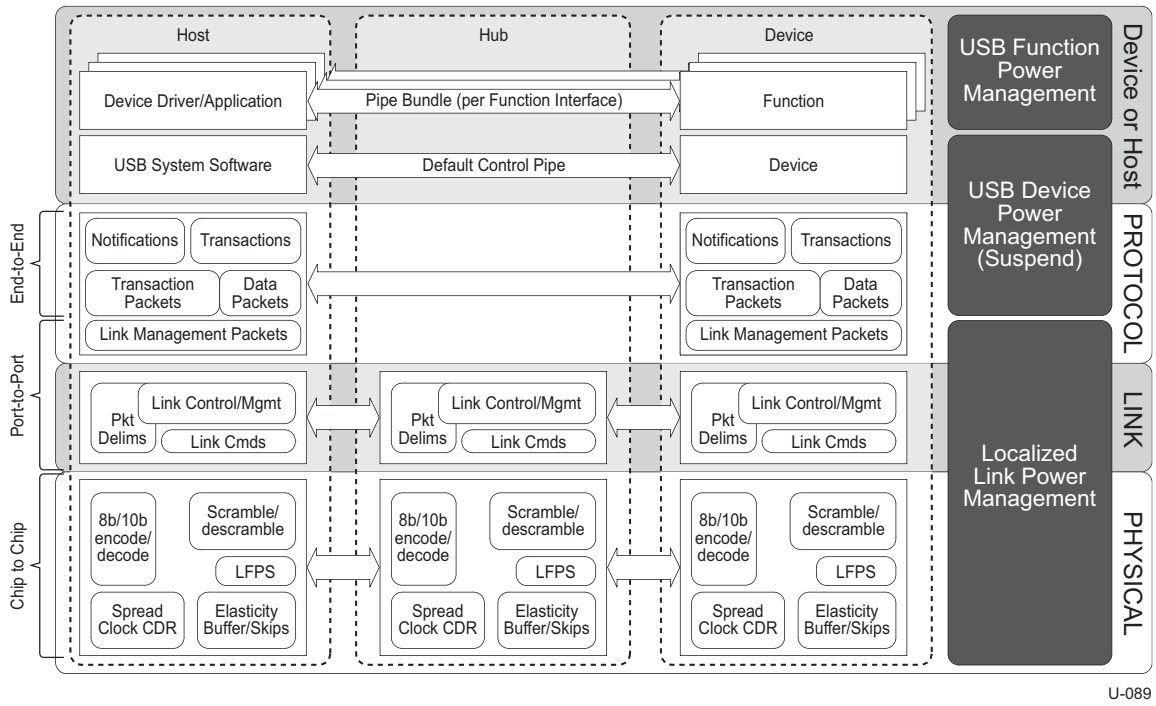


Figure 3-3. SuperSpeed Bus Communications Layers and Power Management Elements

The rows (device or host, protocol, link, physical) realize the communications layers of the SuperSpeed interconnect. Sections 3.2.1 through 3.2.3 provide architectural overviews of each of the communications layers. The three, left-most columns (host, hub, and device) illustrate the topological relationships between devices connected to the SuperSpeed bus; refer to the overview in Sections 3.2.6 through 3.2.7. The right-most column illustrates the influence of power management mechanisms over the communications layers; refer to the overview in Section 3.2.5.

3.2.1 Physical Layer

The physical layer specifications for SuperSpeed are detailed in Chapter 6. The physical layer defines the PHY portion of a port and the physical connection between a downstream facing port (on a host or hub) and the upstream facing port on a device. The SuperSpeed physical connection is comprised of two differential data pairs, one transmit path and one receive path (see Figure 3-2). The nominal signaling data rate is 5 Gbps.

The electrical aspects of each path are characterized as a transmitter, channel, and receiver; these collectively represent a unidirectional differential link. Each differential link is AC-coupled with capacitors located on the transmitter side of the differential link. The channel includes the electrical characteristics of the cables and connectors.

At an electrical level, each differential link is initialized by enabling its receiver termination. The transmitter is responsible for detecting the far end receiver termination as an indication of a bus connection and informing the link layer so the connect status can be factored into link operation and management.

When receiver termination is present but no signaling is occurring on the differential link, it is considered to be in the electrical idle state. When in this state, low frequency periodic signaling (LFPS) is used to signal initialization and power management information. The LFPS is relatively simple to generate and detect and uses very little power.

Each PHY has its own clock domain with Spread Spectrum Clocking (SSC) modulation. The USB 3.0 cable does not include a reference clock so the clock domains on each end of the physical connection are not explicitly connected. Bit-level timing synchronization relies on the local receiver aligning its bit recovery clock to the remote transmitter's clock by phase-locking to the signal transitions in the received bit stream.

The receiver needs enough transitions to reliably recover clock and data from the bit stream. To assure that adequate transitions occur in the bit stream independent of the data content being transmitted, the transmitter encodes data and control characters into symbols using an 8b/10b code. Control symbols are used to achieve byte alignment and are used for framing data and managing the link. Special characteristics make control symbols uniquely identifiable from data symbols.

A number of techniques are employed to improve channel performance. For example, to avoid overdriving and improve eye margin at the receiver, transmitter de-emphasis may be applied when multiple bits of the same polarity are sent. Also, equalization may be used in the receiver with the characteristics of the equalization profile being established adaptively as part of link training.

Signal (timing, jitter tolerance, etc.) and electrical (DC characteristics, channel capacitance, etc.) performance of SuperSpeed links are defined with compliance requirements specified in terms of transmit and receive signaling eyes.

The physical layer receives 8-bit data from the link layer and scrambles the data to reduce EMI emissions. It then encodes the scrambled 8-bit data into 10-bit symbols for transmission over the physical connection. The resultant data are sent at a rate that includes spread spectrum to further lower the EMI emissions. The bit stream is recovered from the differential link by the receiver, assembled into 10-bit symbols, decoded and descrambled, producing 8-bit data that are then sent to the link layer for further processing.

3.2.2 Link Layer

The link layer specifications for SuperSpeed are detailed in Chapter 7. A SuperSpeed link is a logical and physical connection of two ports. The connected ports are called link partners. A port has a physical part (refer to Section 3.2.1) and a logical part. The link layer defines the logical portion of a port and the communications between link partners.

The logical portion of a port has:

- State machines for managing its end of the physical connection. These include physical layer initialization and event management, i.e., connect, removal, and power management.
- State machines and buffering for managing information exchanges with the link partner. It implements protocols for flow control, reliable delivery (port to port) of packet headers, and link power management. The different packet types are defined in Chapter 7.
- Buffering for data and protocol layer information elements.

The logical portion of a port also:

- Provides correct framing of sequences of bytes into packets during transmission; e.g., insertion of packet delimiters

- Detects received packets, including packet delimiters and error checks of received header packets (for reliable delivery)
- Provides an appropriate interface to the protocol layer for pass-through of protocol-layer packet information exchanges

The physical layer provides the logical port an interface through which it is able to:

- Manage the state of its PHY (i.e., its end of the physical connection), including power management and events (connection, removal, and wake).
- Transmit and receive byte streams, with additional signals that qualify the byte stream as control sequences or data. The physical layer includes discrete transmit and receive physical links, therefore, a port is able to simultaneously transmit and receive control and data information.

The protocol between link partners uses specific encoded control sequences. Note that control sequences are encoded to be tolerant to a single bit error. Control sequences are used for port-to-port command protocol, framing of packet data (packet delimiters), etc. There is a link-partner protocol for power management that uses packet headers.

3.2.3 Protocol Layer

The protocol layer specifications for SuperSpeed are detailed in Chapter 8. This protocol layer defines the "end-to-end" communications rules between a host and device (see Figure 3-3).

The SuperSpeed protocol provides for application data information exchanges between a host and a device endpoint. This communications relationship is called a pipe. It is a host-directed protocol, which means the host determines when application data is transferred between the host and device. SuperSpeed is not a polled protocol, as a device is able to asynchronously request service from the host on behalf of a particular endpoint.

All protocol layer communications are accomplished via the exchange of packets. Packets are sequences of data bytes with specific control sequences which serve as delimiters managed by the link layer. Host transmitted protocol packets are routed through intervening hubs directly to a peripheral device. They do not traverse bus paths that are not part of the direct path between the host and the target peripheral device. A peripheral device expects it has been targeted by any protocol layer packet it receives. Device transmitted protocol packets simply flow upstream through hubs to the host.

Packet headers are the building block of the protocol layer. They are fixed size packets with type and subtype field encodings for specific purposes. A small record within a packet header is utilized by the link layer (port-to-port) to manage the flow of the packet from port to port. Packet headers are delivered through the link layer (port-to-port) reliably. The remaining fields are utilized by the end-to-end protocol.

Application data is transmitted within data packet payloads. Data packet payloads are preceded (in the protocol) by a specifically encoded data packet headers. Data packet payloads are not delivered reliably through the link layer (however, the accompanying data packet headers are delivered reliably). The protocol layer supports reliable delivery of data packets via explicit acknowledgement (header) packets and retransmission of lost or corrupt data. Not all data information exchanges utilize data acknowledgements.

Data may be transmitted in bursts of back-to-back sequences of data packets (depending on the scheduling by the host). The protocol allows efficient bus utilization by concurrently transmitting and receiving over the link. For example, a transmitter (host or device) can burst multiple packets

of data back-to-back while the receiver can transmit data acknowledgements without interrupting the burst of data packets. The number of data packets in a specific burst is scheduled by the host. Furthermore, a host may simultaneously schedule multiple OUT bursts to be active at the same time as an IN burst.

The protocol provides flow control support for some transfer types. A device-initiated flow control is signaled by a device via a defined protocol packet. A host-initiated flow control event is realized via the host schedule (host will simply not schedule information flows for a pipe unless it has data or buffering available). On reception of a flow control event, the host will remove the pipe from its schedule. Resumption of scheduling information flows for a pipe may be initiated by the host or device. A device endpoint will notify a host of its readiness (to source or sink data) via an asynchronously transmitted "ready" packet. On reception of the "ready" notification, the host will add the pipe to its schedule, assuming that it still has data or buffering available.

Independent information streams can be explicitly delineated and multiplexed on the bulk transfer type. This means through a single pipe instance, more than one data stream can be tagged by the source and identified by the sink. The protocol provides for the device to direct which data stream is active on the pipe.

Devices may asynchronously transmit notifications to the host. These notifications are used to convey a change in the device or function state.

A host transmits a special packet header to the bus that includes the host's timestamp. The value in this packet is used to keep devices (that need to) in synchronization with the host. In contrast to other packet types, the timestamp packet is forwarded down all paths not in a low power state. The timestamp packet transmission is scheduled by the host at a specification determined period.

3.2.4 Robustness

There are several attributes of SuperSpeed USB that contribute to its robustness:

- Signal integrity using differential drivers, receivers, and shielding
- CRC protection for header and data packets
- Link level header packet retries to ensure their reliable delivery
- End-to-end protocol retries of data packets to ensure their reliable delivery
- Detection of attach and detach and system-level configuration of resources
- Data and control pipe constructs for ensuring independence from adverse interactions between functions

3.2.4.1 Error Detection

The USB SuperSpeed physical layer bit error rate is expected to be less than one in 10^{12} bits. To provide protection against occasional bit errors, packet framing and link commands have sufficient redundancy to tolerate single-bit errors. Each packet includes a CRC to provide error detection of multiple bit errors. When data integrity is required an error recovery procedure may be invoked in hardware or software.

The protocol includes separate CRCs for headers and data packet payloads. Additionally, the link control word (in each packet header) has its own CRC. A failed CRC in the header or link control word is considered a serious error which will result in a link level retry to recover from the error. A failed CRC in a data packet payload is considered to indicate corrupted data and can be handled by the protocol layer with a request to resend the data packet.

The link and physical layers work together to provide reliable packet header transmission. The physical layer provides an error rate that does not exceed (on average) one bit error in every 10^{12} bits. The link layer uses error checking to catch errors and retransmission of the packet header further reducing the packet header error rate.

3.2.4.2 Error Handling

Errors may be handled in hardware or software. Hardware error handling includes reporting and retrying of failed header packets. A USB host controller will try a transmission that encounters errors up to three times before informing the client software of the failure. The client software can recover in an implementation-specific way.

3.2.5 SuperSpeed Power Management

SuperSpeed provides power management at distinct areas in the bus architecture, link, device, and function (refer to Figure 3-3). These power management areas are not tightly coupled but do have dependencies; these mostly deal with allowable power state transitions based on dependencies with power states of links, devices, and functions.

Link power management occurs asynchronously on every link (i.e., locally) in the connected hierarchy. The link power management policy may be driven by the device, the host or a combination of both. The link power state may be driven by the device or by the downstream port inactivity timers that are programmable by host software. The link power states are propagated upwards by hubs (e.g., when all downstream ports are in a low power state, the hub is required to transition its upstream port to a low power state). The decisions to change link power states are made locally. The host does not directly track the individual link power states. Since only those links between the host and device are involved in a given data exchange, links that are not being utilized for data communications can be placed in a lower power state.

The host does not directly control or have visibility of the individual links' power states. This implies that one or more links in the path between the host and device can be in reduced power state when the host initiates a communication on the bus. There are in-band protocol mechanisms that force these links to transition to the operational power state and notify the host that a transition has occurred. The host knows (can calculate) the worst-case transition time to bring a path to any specific device to an active, or ready state, using these mechanisms. Similarly, a device initiating a communication on the bus with its upstream link in a reduced power state, will first transition its link into an operational state which will cause all links between it and the host to transition to the operational state.

The key points of link power management include:

- Devices send asynchronous ready notifications to the host.
- Packets are routed, allowing links that are not involved in data communications to transition to and/or remain in a low power state.
- Packets that encounter ports in low power states cause those ports to transition out of the low power state with indications of the transition event.
- Multiple host or device driven link states with progressively lower power at increased exit latencies.

As with the USB 2.0 bus, devices can be explicitly suspended via a similar port-suspend mechanism. This sets the link to the lowest link power state and sets a limit on the power draw requirement of the device.

SuperSpeed provides support for function power management in addition to device power management. For multi-function (composite) devices, each function can be independently placed into a lower power state. Note that a device will transition into the suspended state when directed by the host via a port command. The device will not automatically transition into the suspended state when all the individual functions within it are suspended.

Functions on devices may be capable of being remote wake sources. The remote-wake feature on a function must be explicitly enabled by the host. Likewise, a protocol notification is available for a function to signal a remote wake event that can be associated with the source function. All remote-wake notifications are functional across all possible combinations of individual link power states on the path between the device and host.

3.2.6 Devices

All SuperSpeed devices share their base architecture with USB 2.0. They are required to carry information for self-identification and generic configuration. They are also required to demonstrate behavior consistent with the defined SuperSpeed Device States.

All devices are assigned a USB address when enumerated by the host. Each device supports one or more pipes through which the host may communicate with the device. All devices must support a designated pipe at endpoint zero to which the device's Default Control Pipe is attached. All devices support a common access mechanism for accessing information through this control pipe. Refer to Chapter 9 for a complete definition of a control pipe.

SuperSpeed inherits the categories of information that are supported on the default control pipe from USB 2.0.

The USB 3.0 specification defines two sets of USB devices that can be connected to a SuperSpeed host. These are described briefly below.

3.2.6.1 Peripheral Devices

A USB 3.0 peripheral device must provide support for both SuperSpeed and at least one other non-SuperSpeed speed. The minimal requirement for non-SuperSpeed is for a device to be detected on a USB 2.0 host and allow system software to direct the user to attach the device to a SuperSpeed capable port. A device implementation may provide appropriate full functionality when operating in non-SuperSpeed mode. Simultaneous operation of SuperSpeed and non-SuperSpeed modes is not allowed for peripheral devices.

USB 3.0 devices within a single physical package (i.e., a single peripheral) can consist of a number of functional topologies including single function, multiple functions on a single peripheral device (composite device), and permanently attached peripheral devices behind an integrated hub (compound device) (see Figure 3-4).

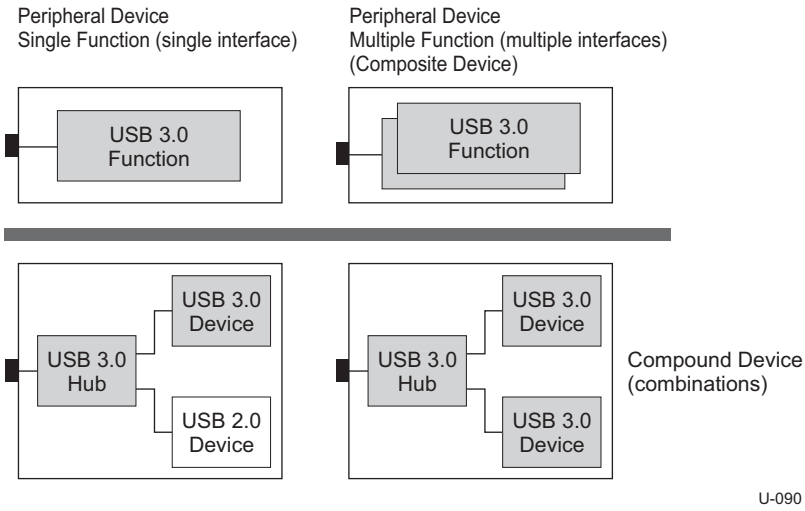


Figure 3-4. Examples of Supported SuperSpeed USB Physical Device Topologies

3.2.6.2 Hubs

The specifications for the SuperSpeed portion of a USB 3.0 hub are detailed in Chapter 10. Hubs have always been a key element in the plug-and-play architecture of the USB. Hosts provide an implementation-specific number of downstream ports to which devices can be attached. Hubs provide additional downstream ports so they provide users with a simple connectivity expansion mechanism for the attachment of additional devices to the USB.

In order to support the dual-bus architecture of USB 3.0, a USB 3.0 hub is the logical combination of two hubs: a USB 2.0 hub and a SuperSpeed hub (see the hub in Figure 3-1). The power and ground from the cable connected to the upstream port are shared across both units within the USB 3.0 hub. The USB 2.0 hub unit is connected to the USB 2.0 data lines and the SuperSpeed hub is connected to the SuperSpeed data lines. A USB 3.0 hub connects upstream as two devices; a SuperSpeed hub on the SuperSpeed bus and a USB 2.0 hub on the USB 2.0 bus.

The SuperSpeed hub manages the SuperSpeed portions of the downstream ports and the USB 2.0 hub manages the USB 2.0 portions of the downstream ports. Each physical port has bus-specific control/status registers. Refer to the *Universal Serial Bus Specification, Revision 2.0* for details on the USB 2.0 hub. Hubs detect device attach, removal, and remote-wake events on downstream ports and enable the distribution of power to downstream devices.

A SuperSpeed hub consists of two logical components: a SuperSpeed hub controller and a SuperSpeed repeater/forwarder. The hub repeater/forwarder is a protocol-controlled router between the SuperSpeed upstream port and downstream ports. It also has hardware support for reset and suspend/resume signaling. The SuperSpeed controller responds to standard, hub-specific status/control commands that are used by a host to configure the hub and to monitor and control its ports.

SuperSpeed hubs actively participate in the (end-to-end) protocol in several ways, including:

- Routes out-bound packets to explicit downstream ports.
- Aggregates in-bound packets to the upstream port.
- Propagates the timestamp packet to all downstream ports not in a low-power state.

- Detects when packets encounter a port that is in a low-power state. The hub transitions the targeted port out of the low-power state and notifies the host and device (in-band) that the packet encountered a port in a low-power state.

3.2.7 Hosts

A USB 3.0 host interacts with USB devices through a host controller. To support the dual-bus architecture of USB 3.0, a host controller must include both SuperSpeed and USB 2.0 elements, which can simultaneously manage control, status and information exchanges between the host and devices over each bus.

The host includes an implementation-specific number of root downstream ports for SuperSpeed and USB 2.0. Through these ports the host:

- Detects the attachment and removal of USB devices
- Manages control flow between the host and USB devices
- Manages data flow between the host and USB devices
- Collects status and activity statistics
- Provides power to attached USB devices

USB System Software inherits its architectural requirements from USB 2.0, including:

- Device enumeration and configuration
- Scheduling of periodic and asynchronous data transfers
- Device and function power management
- Device and bus management information

3.2.8 Data Flow Models

The data flow models for SuperSpeed are described in Chapter 4. SuperSpeed USB inherits the data flow models from USB 2.0, including:

- Data and control exchanges between the host and devices are via sets of either unidirectional or bi-directional pipes.
- Data transfers occur between host software and a particular endpoint on a device. The endpoint is associated with a particular function on the device. These associations between host software to endpoints related to a particular function are called pipes. A device may have more than one active pipe. There are two types of pipes: stream and message. Stream data has no USB-defined structure, while message does. Pipes have associations of data bandwidth, transfer service type (see below), and endpoint characteristics, like direction and buffer size.
- Most pipes come into existence when the device is configured by system software. However, one message pipe, the Default Control Pipe, always exists once a device has been powered and is in the default state, to provide access to the device's configuration, status, and control information.
- A pipe supports one of four transfer types as defined in USB 2.0 (bulk, control, interrupt, and isochronous). The basic architectural elements of these transfer types are unchanged from USB 2.0.
- The bulk transfer type has an extension for SuperSpeed called Streams. Streams provide in-band, protocol-level support for multiplexing multiple independent logical data streams through a standard bulk pipe.

4. SuperSpeed Data Flow Model

This chapter presents a high-level description of how data and information move across the SuperSpeed. Consult the Protocol Layer Chapter for details on the low-level protocol. This chapter provides device framework overview information that is further expanded in the Device Framework Chapter. All implementers should read this chapter to understand the key concepts of SuperSpeed.

4.1 Implementer Viewpoints

SuperSpeed is very similar to USB 2.0 in that it provides communication services between a USB Host and attached USB Devices. The communication model view preserves the USB 2.0 layered architecture and basic components of the communication flow (i.e., point-to-point, same transfer types, etc.). Refer to Chapter 5 in the *Universal Serial Bus Specification, Revision 2.0* for more information about the USB 2.0 communication flow.

This chapter describes the differences (from USB 2.0) of how data and control information is communicated between a SuperSpeed Host and its attached SuperSpeed Devices. In order to understand SuperSpeed data flow, the following concepts are useful:

- Communication Flow Models: Section 4.2 describes how communication flows between the host and devices through the SuperSpeed bus.
- SuperSpeed Protocol Overview: Section 4.3 gives a high level overview of the SuperSpeed protocol and compares it to the USB 2.0 protocol.
- Generalized Transfer Description: Section 4.4 provides an overview of how data transfers work in SuperSpeed and subsequent sections define the operating constraints for each transfer type.
- Device Notifications: Section 4.4.9 provides an overview of Device Notifications, a feature which allows a device to asynchronously notify its host of events or status on the device.
- Reliability and Efficiency: Sections 4.4.10 and 4.4.11 summarize the information and mechanisms available in SuperSpeed to ensure reliability and increase efficiency.

4.2 SuperSpeed Communication Flow

SuperSpeed retains the familiar concepts and mechanisms and support for endpoints, pipes, and transfer types. Refer to the *Universal Serial Bus Specification, Revision 2.0* for details. As in USB 2.0, the ultimate consumer/producer of data is an endpoint.

The endpoint's characteristics (Max Packet Size, Burst Size, etc.) are reported in the endpoint descriptor and the SuperSpeed Endpoint Companion Descriptor. As in USB 2.0, the endpoint is identified using an addressing triple {Device Address, Endpoint Number, Direction}.

All SuperSpeed devices must implement at least the Default Control Pipe (endpoint zero). The Default Control Pipe is a control pipe as defined in the *Universal Serial Bus Specification, Revision 2.0*.

4.2.1 Pipes

A SuperSpeed pipe is an association between an endpoint on a device and software on the host. Pipes represent the ability to move data between software on the host via a memory buffer and an endpoint on a device and have the same behavior as defined in the *Universal Serial Bus Specification, Revision 2.0*. The main difference is that when a non-isochronous endpoint in SuperSpeed is busy it returns a Not Ready (NRDY) response and must send an Endpoint Ready (ERDY) notification when it wants to be serviced again. The host will then reschedule the transaction at the next available opportunity within the constraints of the transfer type.

4.3 SuperSpeed Protocol Overview

As mentioned in the USB 3.0 Architecture Overview Chapter, the SuperSpeed protocol is architected to take advantage of the dual-simplex physical layer. All the USB 2.0 transfer types are supported by the SuperSpeed protocol. The differences between the USB 2.0 protocol and the SuperSpeed protocol are first discussed followed by a brief description of the packets used in SuperSpeed.

4.3.1 Differences from USB 2.0

SuperSpeed is backward compatible with USB 2.0 at the framework level. However, there are some fundamental differences between the USB 2.0 and SuperSpeed protocol:

- USB 2.0 uses a three-part transaction (Token, Data, and Handshake) while SuperSpeed uses the same three parts differently. For OUTs, the token is incorporated in the data packet; while for INs, the Token is replaced by a handshake.
- USB 2.0 does not support bursting while SuperSpeed supports continuous bursting.
- USB 2.0 is a half-duplex broadcast bus while SuperSpeed is a dual-simplex unicast bus which allows concurrent IN and OUT transactions.
- USB 2.0 uses a polling model while SuperSpeed uses asynchronous notifications.
- USB 2.0 does not have a Streaming capability while SuperSpeed supports Streaming for bulk endpoints.
- USB 2.0 offers no mechanism for isochronous capable devices to enter the low power USB bus state between service intervals. SuperSpeed allows isochronous capable devices to autonomously enter low-power link states between service intervals. A SuperSpeed host shall transmit a PING packet to the targeted isochronous device before service interval to allow time for the path to transition back to the active power state before initiating the isochronous transfer.
- USB 2.0 offers no mechanism for device to inform the host how much latency the device can tolerate if the system enters lower system power state. Thus a host may not enter lower system power states as it might impact a device's performance because it lacks an understanding of a device's power policy. USB 3.0 provides a mechanism to allow SuperSpeed devices to inform host of their latency tolerance using Latency Tolerance Messaging. The host may use this information to establish a system power policy that accounts for the devices' latency tolerance.
- USB 2.0 transmits SOF/uSOF at fixed 1 ms/125 µs intervals. A device driver may change the interval with small finite adjustments depending on the implementation of host and system software. USB 3.0 adds mechanism for devices to send a Bus Interval Adjustment Message that is used by the host to adjust its 125 µs bus interval up to +/-13.333 µs. In addition, the host

may send an Isochronous Timestamp Packet (ITP) within a relaxed timing window from a bus interval boundary.

- USB 2.0 power management, including Link Power Management, is always directly initiated by the host. SuperSpeed supports link-level power management that may be initiated from either end of the link. Thus, each link can independently enter low-power states whenever idle and exit whenever communication is needed.
- USB 2.0 handles transaction error detection and recovery and flow control only at the end-to-end level for each transaction. SuperSpeed splits these functions between the end-to-end and link levels.

4.3.1.1 Comparing USB 2.0 and SuperSpeed Transactions

The SuperSpeed dual-simplex physical layer allows information to travel simultaneously in both directions. The SuperSpeed protocol allows the transmitter to send multiple data packets before receiving a handshake. For OUT transfers, the information contained in the USB 2.0 Token is incorporated in the data packet header so a separate Token is not required. For IN transfers, a handshake is sent to the device to request data. The device may respond by either returning data, returning a STALL handshake, or by returning a Not Ready (NRDY) handshake to defer the transfer until the device is ready.

The USB 2.0 broadcasts packets to all enabled downstream ports. Every device is required to decode the address triple {device address, endpoint, and direction} of each packet to determine if it needs to respond. SuperSpeed unicasts the packets; downstream packets are sent over a directed path between the host and the targeted device while upstream packets are sent over the direct path between the device and the host. SuperSpeed packets contain routing information that the hubs use to determine which downstream port the packet needs to traverse to reach the device. There is one exception; the Isochronous Timestamp Packet (ITP) is multicast to all active ports.

USB 2.0 style polling has been replaced with asynchronous notifications. The SuperSpeed transaction is initiated by the host making a request followed by a response from the device. If the device can honor the request, it either accepts or sends data. If the endpoint is halted, the device shall respond with a STALL handshake. If it cannot honor the request due to lack of buffer space or data, it responds with a Not Ready (NRDY) to tell the host that it is not able to process the request at this time. When the device can honor the request, it will send an Endpoint Ready (ERDY) to the host which will then reschedule the transaction.

The move to unicasting and the limited multicasting of packets together with asynchronous notifications allows links that are not actively passing packets to be put into reduced power states. Upstream and downstream ports cooperate to place their link into a reduced power state that hubs will propagate upstream. Allowing link partners to control their independent link power state and a hub's propagating the highest link power state seen on any of its downstream ports to its upstream port, puts the bus into the lowest allowable power state rapidly.

4.3.1.2 Introduction to SuperSpeed Packets

SuperSpeed packets start with a 16-byte header. Some packets consist of a header only. All headers begin with the Packet Type information used to decide how to handle the packet. The header is protected by a 16-bit CRC (CRC-16) and ends with a 2-byte link control word. Depending on the Type, most packets contain routing information (Route String) and a device address triple {device address, endpoint number, and direction}. The Route String is used to direct packets sent by the host on a directed path through the topology. Packets sent by the device are implicitly routed as the hub always forwards a packet seen on any downstream port to its upstream port. There are four basic types of packets: Link Management Packets, Transaction Packets, Data Packets, and Isochronous Timestamp Packets:

- A Link Management Packet (LMP) only traverses a pair of directly connected ports and is primarily used to manage that link.
- A Transaction Packet (TP) traverses all the links in the path directly connecting the host and a device. It is used to control the flow of data packets, configure devices and hubs, etc. Note that a Transaction Packet does not have a data payload.
- A Data Packet (DP) traverses all the links in the path directly connecting the host and a device. Data Packets consist of two parts: a Data Packet Header (DPH) which is similar to a TP and a Data Packet Payload (DPP) which consists of the data block plus a 32-bit CRC (CRC-32) used to ensure the data's integrity.
- An Isochronous Timestamp Packet (ITP) is a multicast packet sent by the host to all active links**.**

4.4 Generalized Transfer Description

Each non-isochronous data packet sent to a receiver is acknowledged by a handshake (called an ACK transaction packet). However, due to the fact that SuperSpeed has independent transmit and receive paths, the transmitter does not have to wait for an explicit handshake for each data packet transferred before sending the next packet.

SuperSpeed preserves all of the basic data flow and transfer concepts defined in USB 2.0, including the transfer types, pipes, and basic data flow model. The differences with USB 2.0 are discussed in this section, starting at the protocol level, followed by transfer type constraints.

The USB 2.0 specification utilizes a serial transaction model. This essentially means that a host starts and completes one bus transaction {Token, Data, Handshake} before starting the next transaction. Split transactions also adhere to this same model since they are comprised of complete high-speed transactions {Token, Data, Handshake} that are completed under the same model as all other transactions.

SuperSpeed improves on the USB 2.0 transaction protocol by using the independent transmit and receive paths. The result is that the SuperSpeed USB transaction protocol is essentially a split-transaction protocol that allows more than one OUT "bus transaction" as well as at most one IN "bus transaction" to be active on the bus at the same time. The order in which a device responds to transactions is fixed on a per endpoint basis (for example, if an endpoint received three DPs, the endpoint must return ACK TPs for each one, in the order that the DPs were received). The order a device responds to ACKs or DPs that are sent to different endpoints on the device is device implementation dependent and software can not expect them to occur/complete in any particular

order. The split-transaction protocol scales well (across multiple transactions to multiple function endpoints) with signaling bit-rates as it is not subject to propagation delays.

The USB 2.0 protocol completes an entire IN or OUT transaction {Token, Data, Handshake} before continuing to the next bus transaction for the next scheduled function endpoint. All transmissions from the host are essentially broadcast on the USB 2.0 bus. In contrast, the SuperSpeed protocol does not broadcast any packets (except for ITPs) and packets traverse only the links needed to reach the intended recipient. The host starts all transactions by sending handshakes or data and devices respond with either data or handshakes. If the device does not have data available or cannot accept the data, it responds with a packet that states that it is not able to do so. Subsequently, when the device is ready to either receive or transmit data it sends a notification to the host that indicates that it is ready to resume transactions. In addition, SuperSpeed provides the ability to transition links into and out of specific low power states. Lower power link states are entered either under software control or under autonomous hardware control after being enabled by software. Mechanisms are provided to automatically transition all links in the path between the host and a device from a non-active power state to the active power state.

Devices report the maximum packet size for each endpoint in its endpoint descriptor. The size indicates data payload length only and does not include any of the overhead for link and protocol level. Bandwidth allocation for SuperSpeed is similar to USB 2.0.

4.4.1 Data Bursting

Data Bursting enhances efficiency by eliminating the wait time for acknowledgements on a per data packet basis. Each endpoint on a SuperSpeed device indicates the number of packets that it can send/receive (called the maximum data burst size) before it has to wait for an explicit handshake. Maximum data burst size is an individual endpoint capability; a host determines an endpoint's maximum data burst size from the SuperSpeed Endpoint Companion descriptor associated with this endpoint (refer to Section 9.6.7).

The host may dynamically change the burst size on a per-transaction basis up to the configured maximum burst size. Examples of when a host may use different burst sizes include, but are not limited to, a fairness policy on the host and retries for an interrupt stream. When the endpoint is an OUT, the host can easily control the burst size (the receiver must always be able to manage a transaction burst size). When the endpoint is an IN, the host can limit the burst size for the endpoint on a per-transaction basis via a field in the acknowledgement packet sent to the device.

4.4.2 IN Transfers

The host and device shall adhere to the constraints of the transfer type and endpoint characteristics.

A host initiates a transfer by sending an acknowledgement packet (IN) to the device. This acknowledgement packet contains the addressing information required to route the packet to the intended endpoint. The host tells the device the number of data packets it can send and the sequence number of the first data packet expected from the device. In response the endpoint will transmit data packet(s) with the appropriate sequence numbers back to the host. The acknowledgement packet also implicitly acknowledges the previous data packet that was received successfully.

Note that even though the host is required to send an acknowledgement packet for every data packet received, the device can send up to the number of data packets requested without waiting for any acknowledgement packet.

The SuperSpeed IN transaction protocol is illustrated in Figure 4-1. An IN transfer on the SuperSpeed bus consists of one or more IN transactions consisting of one or more packets and completes when any one of the following conditions occurs:

- All the data for the transfer is successfully received.
- The endpoint responds with a packet that is less than the endpoint's maximum packet size.
- The endpoint responds with an error.

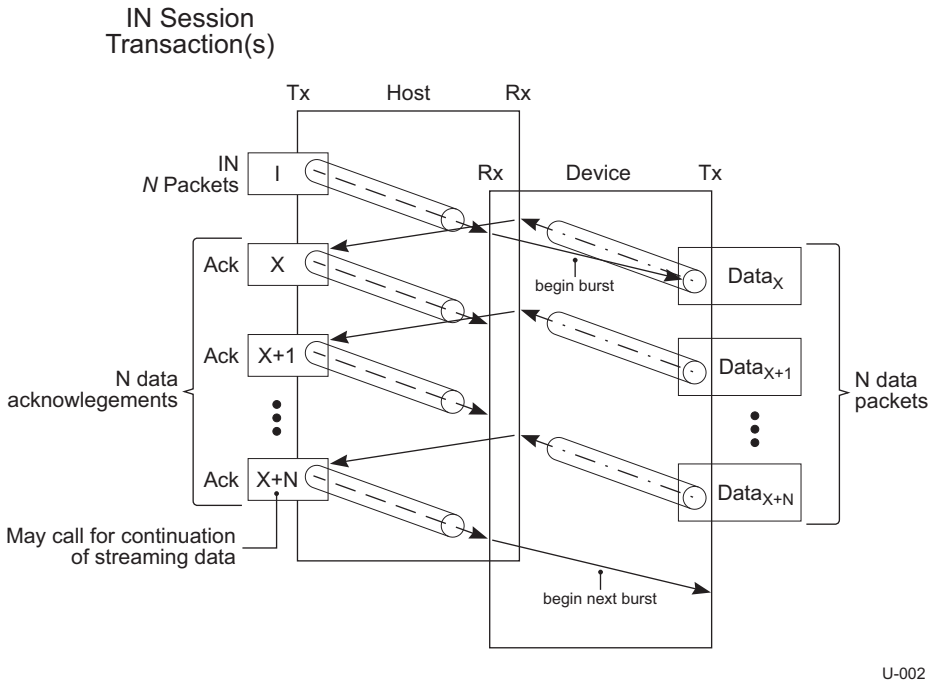


Figure 4-1. SuperSpeed IN Transaction Protocol

4.4.3 OUT Transfers

The host and device shall adhere to the constraints of the transfer type and endpoint characteristics.

A host initiates a transfer by sending a burst of data packets to the device. Each data packet contains the addressing information required to route the packet to the intended endpoint. It also includes the sequence number of the data packet. For a non-isochronous transaction, the device returns an acknowledgement packet including the sequence number for the next data packet and implicitly acknowledging the current data packet.

Note that even though the device is required to send an acknowledgement packet for every data packet received, the host can send up to the maximum burst size number of data packets to the device without waiting for an acknowledgement.

The SuperSpeed OUT transaction protocol is illustrated in Figure 4-2. An OUT transfer on the SuperSpeed bus consists of one or more OUT transactions consisting of one or more packets and completes when any one of the following conditions occurs:

- All the data for the transfer is successfully transmitted.
- The host sends a packet that is less than the endpoints maximum packet size.
- The endpoint responds with an error.

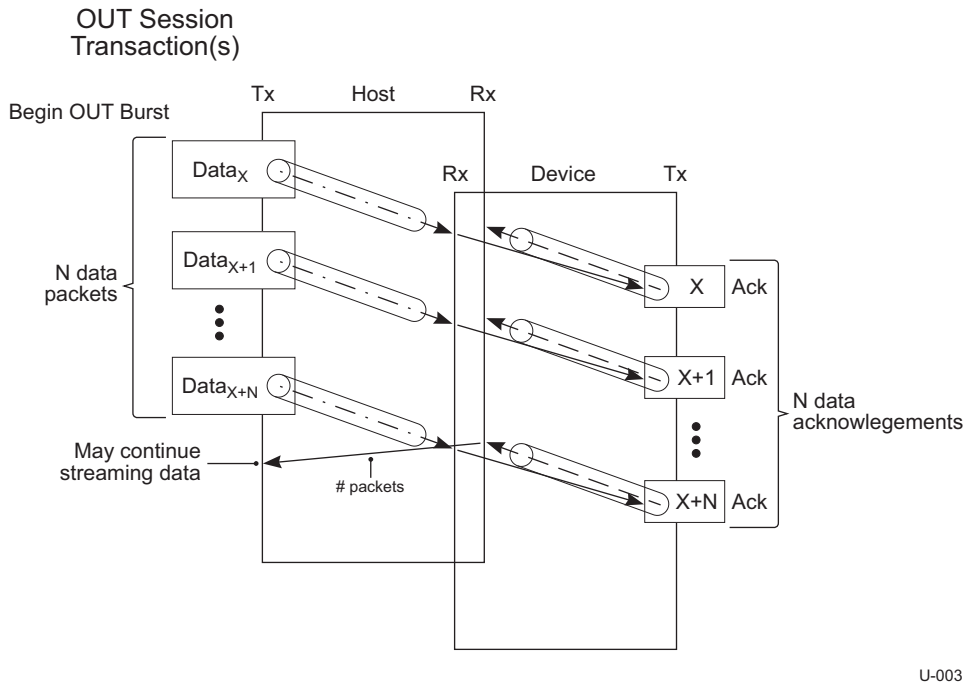


Figure 4-2. SuperSpeed OUT Transaction Protocol

4.4.4 Power Management and Performance

The use of inactivity timers and device-driven link power management provides the ability for very aggressive power management. When the host sends a packet to a device behind a hub with a port whose link is in a non-active state, the packet will not be able to traverse the link until it returns to the active state. In the case of an IN transaction, the host will not be able to start another IN transaction until the current one completes. The effect of this behavior could have a significant impact on overall performance.

To balance power management with good performance, the concept of a deferral (to both INs and OUTs) is used. When a host initiates a transaction that encounters a link in a non-active state, a deferred response is sent by the hub to tell the host that this particular path is in a reduced power managed state and that the host should go on to schedule other transactions. In addition, the hub sends a deferred request to the device to notify it that a transaction was attempted. This mechanism informs the host of added latency due to power management and allows the host to mitigate performance impacts that result from the link power management.

4.4.5 Control Transfers

The purpose and characteristics of Control Transfers are identical to those defined in Section 5.5 of the *Universal Serial Bus Specification, Revision 2.0*. The Protocol Layer chapter of this specification describes the details of the packets, bus transactions, and transaction sequences used to accomplish Control transfers. The Device Framework chapter of this specification defines the complete set of standard command codes used for devices.

Each device is required to implement the default control pipe as a message pipe. This pipe is intended for device initialization and management. This pipe is used to access device descriptors and to make requests of the device to manipulate its behavior (at a device-level). Control transfers must adhere to the same request definitions described in the *Universal Serial Bus Specification, Revision 2.0*.

The SuperSpeed system will make a "best effort" to support delivery of control transfers between the host and devices. As with USB 2.0, a function and its client software cannot request specific bandwidth for control transfers.

4.4.5.1 Control Transfer Packet Size

Control endpoints have a fixed maximum control transfer data payload size of 512 bytes and have a maximum burst size of one. These maximums apply to all data transactions during the data stage of the control transfer. Refer to Section 8.12.2 for detailed information on the Setup and Status stages of a control transfer in SuperSpeed.

A SuperSpeed device must report a value of 09H in the bMaxPacketSize field of its Device Descriptor. The rule for decoding the default maximum packet size for the Default Control Pipe is given in Section 9.6.1. The Default Control Pipe must support a maximum sequence value of 32 (i.e., sequence values in the range [0-31] are used).

The requirements for data delivery and completion of device-to-host and host-to-device Data stages are generally not changed between USB 2.0 and SuperSpeed (refer to Section 5.5.3 of the *Universal Serial Bus Specification, Revision 2.0*).

4.4.5.2 Control Transfer Bandwidth Requirements

A device has no way to indicate the desired bandwidth for a control pipe. A host balances the bus access requirements of all control pipes and pending transactions on those pipes to provide a "best effort" delivery between client software and functions on the device. This policy is the same as the USB 2.0 policy.

SuperSpeed requires that bus bandwidth be reserved to be available for use by control transfers as follows:

- The transactions of a control transfer may be scheduled coincident with transactions for other function endpoints of any defined transfer type.
- Retries of control transfers are not given priority over other best effort transactions.
- If there are control and bulk transfers pending for multiple endpoints, control transfers for different endpoints are selected for service according to a fair access policy that is host controller implementation-dependent.

- When a control endpoint delivers a flow control event (as defined in Section 8.10.1), the host will remove the endpoint from the actively scheduled endpoints. The host will resume the transfer to the endpoint upon receipt of a ready notification from the device.

These requirements allow control transfers between a host and devices to regularly move data across the SuperSpeed bus with "best effort." System software's discretionary behavior defined in Section 5.5.4 of the *Universal Serial Bus Specification, Revision 2.0* applies equally to SuperSpeed control transfers.

4.4.5.3 Control Transfer Data Sequences

SuperSpeed preserves the message format and general stage sequencing of control transfers defined in Section 5.5.5 of the *Universal Serial Bus Specification, Revision 2.0*. The SuperSpeed protocol defines some changes to the Setup and Status stages of a control transfer. However, all of the sequencing requirements for normal and error recovery scenarios defined in Section 5.5.5 of the *Universal Serial Bus Specification, Revision 2.0* directly map to the SuperSpeed Protocol.

4.4.6 Bulk Transfers

The purpose and characteristics of Bulk Transfers are similar to those defined in Section 5.8 of the *Universal Serial Bus Specification, Revision 2.0*. Section 8.12.1 of this specification describes the details of the packets, bus transactions and transaction sequences used to accomplish Bulk transfers. The Bulk transfer type is intended to support devices that want to communicate relatively large amounts of data at highly variable times where the transfer can use any available SuperSpeed bandwidth. A SuperSpeed Bulk function endpoint provides the following:

- Access to the SuperSpeed bus on a bandwidth available basis
- Guaranteed delivery of data, but no guarantee of bandwidth or latency

SuperSpeed retains the following characteristics of bulk pipes:

- No data content structure is imposed on the communication flow for bulk pipes.
- A bulk pipe is a stream pipe and, therefore, always has communication flow either into or out of the host for any pipe instance. If an application requires a bi-directional bulk communication flow, two bulk pipes must be used (one IN and one OUT).

Standard USB bulk pipes provide the ability to move a stream of data. SuperSpeed adds the concept of Streams that provide protocol-level support for a multi-stream model.

4.4.6.1 Bulk Transfer Data Packet Size

An endpoint for bulk transfers shall set the maximum data packet payload size in its endpoint descriptor to 1024 bytes. It also specifies the burst size that the endpoint can accept from or transmit on the SuperSpeed bus. The allowable burst size for a bulk endpoint shall be in the range of 1 to 16. All SuperSpeed bulk endpoints shall support sequence values in the range [0-31].

A host is required to support any SuperSpeed bulk endpoint. A host shall support all bulk burst sizes. The host ensures that no data payload of any data packet in a burst transaction will be sent to the endpoint that is larger than the maximum packet size. Additionally, it will not send more data packets than the reported maximum burst size.

A bulk function endpoint must always transmit data payloads with data fields less than or equal to 1024 bytes. If the bulk transfer has more data than that, all data payloads in the burst transaction

are required to be 1024 bytes in length except for the last data payload in the burst, which may contain the remaining data. A bulk transfer may span multiple bus transactions. A bulk transfer is complete when the endpoint does one of the following:

- Has transferred exactly the amount of data expected.
- Transfers a data packet with a payload less than 1024 bytes.
- Responds with a STALL handshake.

4.4.6.2 Bulk Transfer Bandwidth Requirements

As with USB 2.0 a bulk function endpoint has no way to indicate a desired bandwidth for a bulk pipe. Bulk transactions occur on the SuperSpeed bus only on a bandwidth available basis. SuperSpeed provides a "good effort" delivery of bulk data between client software and device functions. Moving control transfers over the SuperSpeed bus has priority over moving bulk transactions. When there are bulk transfers pending for multiple endpoints, the host will provide transaction opportunities to individual endpoints according to a fair access policy, which is host implementation dependent.

All bulk transfers pending in a system contend for the same available bus time. An endpoint and its client software cannot assume a specific rate of service for bulk transfers. Bus time made available to a software client and its endpoint can be changed as other devices are inserted into and removed from the system or as bulk transfers are requested for other function endpoints. Client software cannot assume ordering between bulk and control transfers; i.e., in some situations, bulk transfers can be delivered ahead of control transfers.

The host can use any burst size between 1 and the reported maximum in transactions with a bulk endpoint to more effectively utilize the available bandwidth. For example, there may be more bulk transfers than bandwidth available, so a host can employ a policy of using smaller data bursts per transactions to provide fair service to all pending bulk data streams.

When a bulk endpoint delivers a flow control event (as defined in Section 8.10.1), the host will remove it from the actively scheduled endpoints. The host will resume the transfer to the endpoint upon receipt of a ready notification from the device.

4.4.6.3 Bulk Transfer Data Sequences

Bulk transactions use the standard burst sequence for reliable data delivery defined in Section 8.10.2. Bulk endpoints are initialized to the initial transmit or receive sequence number and burst size (refer to Section 8.12.1.2 and Section 8.12.1.3) by an appropriate control transfer (SetConfiguration, SetInterface, ClearEndpointFeature). Likewise, a host assumes the initial transmit or receive sequence number and burst size for bulk pipes after it has successfully completed the appropriate control transfer as mentioned above.

Halt conditions for a SuperSpeed bulk pipe have the identical side effects as defined for a USB 2.0 bulk endpoint. Recovery from halt conditions are also identical to USB 2.0 (refer to Section 5.8.5 of the *Universal Serial Bus Specification, Revision 2.0).* A bulk pipe halt condition includes a STALL handshake response to a transaction or exhaustion of the host's transaction retry policy due to transmission errors.

4.4.6.4 Bulk Streams

A standard USB Bulk Pipe represents the ability to move single stream of (FIFO) data between the host and a device via a host memory buffer and a device endpoint. SuperSpeed streams provide protocol-level support for a multi-stream model and utilize the "stream" pipe communications mode (refer to Section 5.3.2 of the *Universal Serial Bus Specification, Revision 2.0)*.

Streams are managed between the host and a device using the *Stream Protocol*. Each Stream is assigned a *Stream ID* (SID).

The Stream Protocol defines a handshake, which allows the device or host to establish the *Current Stream* (CStream) ID associated with an endpoint. The host uses the CStream ID to select the command or operation-specific Endpoint Buffer(s) that will be used for subsequent data transfers on the pipe (see Figure 4-3). The device uses the CStream ID to select the Function Data buffer(s) that will be used.

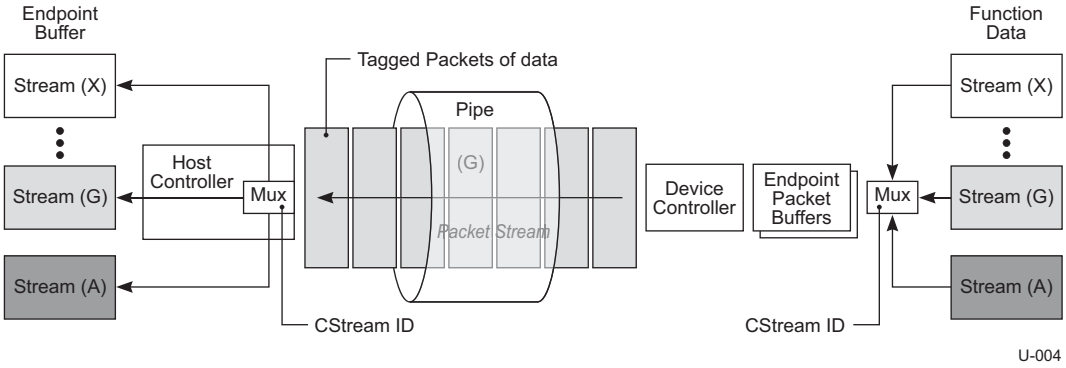


Figure 4-3. USB SuperSpeed IN Stream Example

The example in Figure 4-3 represents an IN Bulk pipe, where a large number of Streams have been established. Associated with each Stream in host memory is one or more Endpoint Buffers to receive the Stream data. In the device, there is a corresponding command or operation-specific Function Data to be transmitted to the host.

When the device has data available for a specific Stream (G in this example), it issues an ERDY tagged with the CStream ID, and the host will begin issuing IN ACK TP's to the device that is tagged with the CStream ID. The device will respond by returning DPs that contain the Function Data associated with the CStream ID that is also tagged with the CStream ID. When the host receives the data, it uses the CStream ID to select the set of Endpoint Buffers that will receive the data.

When the Function Data is exhausted, the device terminates the Stream (refer to Section 8.12.1.4). The host is also allowed to terminate the Stream if it runs out of Endpoint Buffer space.

Streams may be used, for example, to support out-of-order data transfers required for mass storage device command queuing.

A standard bulk endpoint has a single set of Endpoint Buffers associated with it. Streams extend the number of host buffers accessible by an endpoint from 1 to up to 65533. There is a 1:1 mapping between a host buffer and a Stream ID.

Device Class defined methods are used for coordinating the Stream IDs that are used by the host to select Endpoint Buffers and the device to select Function Data associated with a particular Stream. Typically this is done via an out-of-band mechanism (e.g., another endpoint) that is used to pass the list of valid Stream IDs between the host and the device.

The selection of the Current Stream may be initiated by the host or the device and, in either case, the Stream Protocol provides a method for a selection to be rejected. For example, the host may reject a Stream selection initiated by the device if it has no Endpoint Buffers available for it. Or the device may reject a Stream selection initiated by the host if it has no Function Data available for it. The Device Class defines when a stream may be selected by the host or the device, and the actions that will be taken when a Stream is rejected (refer to Section 8.12.1.4).

A combination of vendor and Device Class defined algorithms determine how Streams are scheduled by a device. The Stream protocol provides methods for starting, stopping, and switching Streams (refer to Section 8.12.1.4).

Mechanisms defined by the Stream protocol allow the device or the host to flow control a Stream. These mechanisms overlap with the standard bulk flow control mechanism.

The host also may start or stop a Stream. For instance, the host will stop a Stream if it runs out of buffer space for the Stream. When the host controller informs the device of this condition, the device may switch to another Stream or wait and continue the same Stream when the host receives more buffers.

The Stream Protocol also provides a mechanism which allows the host to asynchronously inform the device when Endpoint Buffers have been added to the pipe. This is useful in cases in which the host must terminate a stream because it ran out of Endpoint Buffers; however the device still has more Function Data to transfer. Without this mechanism, the device would have to periodically retry starting the Stream (impacting power management), or a long latency out-of-band method would be required.

Since Streams are run over a standard bulk pipe, an error will halt the pipe, stopping all stream activity. Removal of the halt condition is achieved via software intervention through a separate control pipe as it is for a standard bulk pipe.

Finally, Streams significantly increase the functionality of a bulk endpoint, while having a minimal impact on the additional hardware required to support the feature in hosts and devices.

4.4.7 Interrupt Transfers

The purpose and characteristics of interrupt transfers are similar to those defined in USB 2.0 (see Section 5.7 of the *Universal Serial Bus Specification, Revision 2.0*). The SuperSpeed interrupt transfer types are intended to support devices that require a high reliability method to communicate a small amount of data with a bounded service interval. The Protocol Layer chapter of this specification describes the details of the packets, bus transactions and transaction sequences used to accomplish Interrupt transfers. The SuperSpeed Interrupt transfer type nominally provides the following:

- Guaranteed maximum service interval
- Guaranteed retry of transfer attempts in the next service interval

Interrupt transfers are attempted each service interval for an interrupt endpoint. Bandwidth is reserved to guarantee a transfer attempt each service interval. Once a transfer is successful, another

transfer attempt is not made until the next service interval. If the endpoint responds with a not ready notification or an acknowledgement indicating that it cannot accept any more packets, the host will not attempt another transfer to that endpoint until it receives a ready notification. The host must then service the endpoint within the larger of (a) twice the service interval, and (b) the device's last reported BELT, after receipt of the ready notification. The requested service interval for the endpoint is described in its endpoint descriptor.

SuperSpeed retains the following characteristics of interrupt pipes:

- No data content structure is imposed on communication flow for interrupt pipes
- An interrupt pipe is a stream pipe and, therefore, is always unidirectional

4.4.7.1 Interrupt Transfer Packet Size

An endpoint for interrupt transfers specifies the maximum data packet payload size that it can accept from or transmit on the SuperSpeed bus. The only allowable maximum data payload size for interrupt endpoints is 1024 bytes for interrupt endpoints that support a burst size greater than one and can be any size from 1 to 1024 for an interrupt endpoint with a burst size equal to one. The maximum allowable burst size for interrupt endpoints is three. All SuperSpeed interrupt endpoints shall support sequence values in the range [0-31].

SuperSpeed interrupt endpoints are only intended for moving small amounts of data with a bounded service interval. The SuperSpeed protocol does not require the interrupt transactions to be maximum size.

A host is required to support SuperSpeed interrupt endpoints. A host shall support all allowed combinations of interrupt packet sizes and burst sizes. The host ensures that no data payload of any data packet in a burst transaction shall be sent to the endpoint that is larger than the endpoint's maximum packet size. Also, the host shall not send more data packets in a burst transaction than the endpoint's maximum burst size.

An interrupt endpoint shall always transmit data payloads with data fields less than or equal to the endpoint's maximum packet size. If the interrupt transfer has more information than will fit into the maximum packet size for the endpoint, all data payloads in the burst transaction are required to be maximum packet size except for the last data payload in the burst transaction, which may contain the remaining data. An interrupt transfer may span multiple burst transactions.

An interrupt transfer is complete when the endpoint does one of the following:

- Has transferred exactly the amount of data expected
- Transfers a data packet with a payload less than the maximum packet size
- Responds with a STALL handshake

4.4.7.2 Interrupt Transfer Bandwidth Requirements

Periodic endpoints may be allocated up to 90% of the total available bandwidth on SuperSpeed.

An endpoint for an interrupt pipe specifies its desired service interval bound via its endpoint descriptor. An interrupt endpoint can specify a desired period $2^{(bInterval-1)}$ x 125 μs, where bInterval is in the range 1 up to (and including) 16. The USB System Software will use this information during configuration to determine a period that can be sustained. The period provided by the system may be shorter than that desired by the device up to the shortest period defined by the SuperSpeed (125 μs which is also referred to as a bus interval). Note that errors on the bus can

prevent an interrupt transaction from being successfully delivered over the bus and consequently exceed the desired period.

A SuperSpeed interrupt endpoint can move up to three maximum sized packets (3 x 1024 bytes) per service interval. Interrupt transfers are moved over the USB by accessing an interrupt endpoint every service interval. For interrupt endpoints, the host has no way to determine whether the endpoint will source/sync data without accessing the endpoint and requesting an interrupt transfer. If an interrupt IN endpoint has no interrupt data to transmit or an interrupt OUT endpoint has insufficient buffer to accept data when accessed by the host, it responds with a flow control response.

An endpoint should only provide interrupt data when it has interrupt data pending to avoid having a software client erroneously notified of a transfer completion. A zero-length data payload is a valid transfer and may be useful for some implementations. The host may access an endpoint at any point during the service interval. The interrupt endpoint should not assume a fixed spacing between transaction attempts. The interrupt endpoint can assume only that it will receive a transaction attempt within the service interval bound. Errors can prevent the successful exchange of data within the service interval bound and a host is not required to retry the transaction in the same service interval and is only required to retry the transaction in the next service interval.

4.4.7.3 Interrupt Transfer Data Sequences

Interrupt transactions use the standard burst sequence for reliable data delivery protocol defined in Section 8.10.2. Interrupt endpoints are initialized to the initial transmit or receive sequence number and burst size (refer to Section 8.12.4.1 and Section 8.12.4.2) by an appropriate control transfer (SetConfiguration, SetInterface, ClearEndpointFeature). A host sets the initial transmit or receive sequence number and burst size for interrupt pipes after it has successfully completed the appropriate control transfer.

Halt conditions for a SuperSpeed interrupt pipe have the identical side effects as defined for a USB 2.0 interrupt endpoint. Recovery from halt conditions are also identical to the USB 2.0, refer to Section 5.7.5 in the *Universal Serial Bus Specification, Revision 2.0*. An interrupt pipe halt condition includes a STALL handshake response to a transaction or exhaustion of the host's transaction retry policy due to transmission errors.

4.4.8 Isochronous Transfers

The purpose of SuperSpeed isochronous transfers is similar to those defined in USB 2.0 (refer to Section 5.6 of the *Universal Serial Bus Specification, Revision 2.0*). As in USB 2.0, the SuperSpeed isochronous transfer type is intended to support streams that want to perform error tolerant, periodic transfers within a bounded service interval. SuperSpeed does not transmit start of frames as on USB 2.0, but timing information is transmitted to devices via Isochronous Timestamp Packets (ITPs). The Protocol Layer chapter of this specification describes the details of the packets, bus transactions, and transaction sequences used to accomplish isochronous transfers. It also describes how the timing information is conveyed to devices. The SuperSpeed isochronous transfer type provides the following:

- Guaranteed bandwidth for transaction attempts on the SuperSpeed bus with bounded latency
- Guaranteed data rate through the pipe as long as data is provided to the pipe

Isochronous transactions are attempted each service interval for an isochronous endpoint. Isochronous endpoints that are admitted on the SuperSpeed bus are guaranteed the bandwidth they require on the bus. The host can request data from the device or send data to the device at any time during the service interval for a particular endpoint on that device. The requested service interval for the endpoint is described in its endpoint descriptor. The SuperSpeed isochronous transfer type is designed to support a source and sink that produce and consume data at the same average rate.

A SuperSpeed isochronous pipe is a stream pipe and is always unidirectional. The endpoint description identifies whether a given isochronous pipe's communication flow is into or out of the host. If a device requires bi-directional isochronous communication flows, two isochronous pipes must be used, one in each direction.

SuperSpeed power management may interfere with isochronous transfers whenever an isochronous transfer needs to traverse a non-active link. The resultant delay could result in the data not arriving within the service interval. To overcome this, SuperSpeed defines a PING and PING_RESPONSE mechanism (refer to Section 8.5.7). Before initiating an isochronous transfer the host shall send a PING packet to the device. The device responds with a PING_RESPONSE packet that tells the host that all the links in the path to the device are in the active state.

4.4.8.1 Isochronous Transfer Packet Size

An endpoint for isochronous transfers specifies the maximum data packet payload size that the endpoint can accept from or transmit on SuperSpeed. The only allowable maximum data payload size for isochronous endpoints is 1024 bytes for isochronous endpoints that support a burst size greater than one and can be any size from 0 to 1024 for an isochronous endpoint with a burst size equal to one. The maximum allowable burst size for isochronous endpoints is 16. However an isochronous endpoint can request up to three burst transactions in the same service interval.

The SuperSpeed protocol does not require the isochronous data packets to be maximum size. If an amount of data less than the maximum packet size is being transferred, the data packet shall not be padded.

A host shall support SuperSpeed isochronous endpoints for all allowed combinations of isochronous packet sizes and burst sizes. The host shall ensure that no data payload of any data packet in a burst transaction be sent to the endpoint that is larger than the reported maximum packet size. Also, the host shall not send more data packets in a burst transaction than the endpoint's maximum burst size.

An isochronous endpoint shall always transmit data payloads with data fields less than or equal to the endpoint's maximum packet size. If the isochronous transfer has more information than will fit into the maximum packet size for the endpoint, all data payloads in the burst transaction are required to be maximum packet size except for the last data payload in the burst transaction, which may contain the remaining data. An isochronous transfer may span multiple burst transactions.

4.4.8.2 Isochronous Transfer Bandwidth Requirements

Periodic endpoints can be allocated up to 90% of the total available bandwidth on SuperSpeed.

An endpoint for an isochronous pipe specifies its desired service interval bound via its endpoint descriptor. An isochronous endpoint can specify a desired period $2^{(bInterval-1)}$ x 125 μs, where bInterval is in the range 1 to 16. The system software will use this information during

configuration to determine whether the endpoint can be added to the host schedule. Note that errors on the bus can prevent an isochronous transaction from being successfully delivered over the bus.

A SuperSpeed isochronous endpoint can move up to three burst transactions of up to 16 maximum sized packets (3 x 16 x 1024 bytes) per service interval. Isochronous transfers are moved over the USB by accessing an isochronous endpoint every service interval. The host will send data or request data to or from the endpoint every service interval. Note, if an endpoint has no isochronous data to transmit when accessed by the host, it shall send a zero length packet in response to the request for data.

The host may access an endpoint at any point during the appropriate service interval. The isochronous endpoint should not assume a fixed spacing between transaction attempts. The isochronous endpoint can assume only that it will receive a transaction attempt within the service interval bound. Errors may prevent the successful exchange of data within the service interval bound, however since the packets in an isochronous transaction are not acknowledged, a host has no way of knowing which packets were not received successfully and hence will not retry packets.

4.4.8.3 Isochronous Transfer Data Sequences

Isochronous endpoints always transmit data packets starting with sequence number zero in each service interval. Each successive data packet transmitted in the same service interval is sent with the next higher sequence number. The sequence number shall roll over from thirty one to zero when transmitting the thirty second packet. Isochronous endpoints do not support retries and cannot respond with flow control responses.

4.4.8.4 Special Considerations for Isochronous Transfers

For a general overview of isochronous data movements over USB, USB clock model, clock synchronization, and the different types of USB-defined synchronization types and their specific requirements, refer to the *USB 2.0 Specification*, Section 5.12. The following section presents the information necessary to implement SuperSpeed isochronous endpoints that need an explicit feedback isochronous endpoint.

4.4.8.4.1 Explicit Feedback

A SuperSpeed asynchronous isochronous sink endpoint must provide explicit feedback to the host by indicating accurately what its desired data rate (F_f) is, relative to the USB bus interval frequency. This allows the host to continuously adjust the number of samples sent to the sink so that neither underflow nor overflow of the data buffer occurs. Likewise, a SuperSpeed adaptive source endpoint must receive explicit feedback from the host so that it can accurately generate the number of samples required by the host. Feedback endpoints can be specified as described in Section 9.6.6 for the *bmAttributes* field of the endpoint descriptor.

To generate the desired data rate F_f, the device must measure its actual sampling rate F_s, referenced to the USB notion of time, i.e., the USB bus interval frequency. This specification requires the data rate F_f to be resolved to better than one sample per second (1 Hz) in order to allow a high-quality source rate to be created and to tolerate delays and errors in the feedback loop. To achieve this accuracy, the measurement time T_{meas} must be at least 1 second. Therefore:

$$T_{meas} = 2^K$$

where T_{meas} is now expressed in USB bus intervals and $K \geq 13$ for SuperSpeed devices (125 µs bus intervals). However, in most devices, the actual sampling rate F_s is derived from a master clock F_m through a binary divider. Therefore:

$$F_m = F_s * 2^P$$

where P is a positive integer (including 0 if no higher-frequency master clock is available). The measurement time T_{meas} can now be decreased by measuring F_m instead of F_s and:

$$T_{meas} = \frac{2^K}{2^P} = 2^{(K-P)}$$

In this way, a new estimate for F_f becomes available every $2^{(K-P)}$ bus intervals. P is practically bound to be in the range [0,K] because there is no point in using a clock slower than F_s (P=0), and no point in trying to update F_f more than once per bus interval (P=K). A sink can determine F_f by counting cycles of the master clock F_m for a period of $2^{(K-P)}$ bus intervals. The counter is read into F_f and reset every $2^{(K-P)}$ bus intervals. As long as no clock cycles are skipped, the count will be accurate over the long term.

Each bus interval, an adaptive source adds F_f to any remaining fractional sample count from the previous bus interval, sources the number of samples in the integer part of the sum, and retains the fractional sample count for the next bus interval. The source can look at the behavior of F_f over many bus intervals to determine an even more accurate rate, if it needs to.

F_f is expressed in number of samples per bus interval. The F_f value consists of an integer part that represents the (integer) number of samples per bus interval and a fractional part that represents the "fraction" of a sample that would be needed to match the sampling frequency F_s to a resolution of 1 Hz or better. The fractional part requires at least K bits to represent the "fraction" of a sample to a resolution of 1 Hz or better. The integer part must have enough bits to represent the maximum number of samples that can ever occur in a single bus interval. Assuming that the minimum sample size is one byte, then this number is currently limited to 48*1024=49152 and 16 bits are needed.

For SuperSpeed endpoints, the F_f value shall be encoded in an unsigned 32.K ($K \geq 13$) format, encoded into eight bytes (for future extensibility). The value shall be aligned into these eight bytes so that the binary point is located between the fourth and the fifth byte so that it has a 32.32 format. Only the first K bits behind the binary point are required. The lower 32-K bits may be optionally used to extend the precision of F_f, otherwise, they shall be reported as zero.

An endpoint needs to implement only the number of bits that it effectively requires for its maximum F_f.

The choice of P is endpoint-specific. Use the following guidelines when choosing P:

- P must be in the range [0,K].
- Larger values of P are preferred, because they reduce the size of the frame counter and increase the rate at which F_f is updated. More frequent updates result in a tighter control of the source data rate, which reduces the buffer space required to handle F_f changes.
- P should be less than K so that F_f is averaged across at least two frames in order to reduce SOF jitter effects.
- P should not be zero in order to keep the deviation in the number of samples sourced to less than 1 in the event of a lost F_f value.

Isochronous transfers are used to read F_f from the feedback register. The desired reporting rate for the feedback should be $2^{(K-P)}$ bus intervals. F_f will be reported at most once per update period. There is nothing to be gained by reporting the same F_f value more than once per update period. The endpoint may choose to report F_f only if the updated value has changed from the previous F_f value. If the value has not changed, the endpoint may report the current F_f value or a zero length data payload. It is strongly recommended that an endpoint always report the current F_f value any time it is polled.

It is possible that the source will deliver one too many or one too few samples over a long period due to errors or accumulated inaccuracies in measuring F_f. The sink must have sufficient buffer capability to accommodate this. When the sink recognizes this condition, it should adjust the reported F_f value to correct it. This may also be necessary to compensate for relative clock drifts. The implementation of this correction process is endpoint-specific and is not specified.

4.4.9 Device Notifications

Device notifications are a standard method for a device to communicate asynchronous device- and bus-level event information to the host. This feature does not map to the pipe model defined for the standard transfer types. Device notifications are always initiated by a device and the flow of data information is always device to host.

Device notifications are message-oriented data communications that have a specific data format structure as defined in Section 8.5.6. Device notifications do not have any data payload. Devices can send a device notification at any time.

4.4.10 Reliability

To ensure reliable operation, several layers of protection are used. This provides reliability for both flow control and data end to end.

4.4.10.1 Physical Layer

The SuperSpeed physical layer provides bit error rates less than 1 bit in 10^{12} bits.

4.4.10.2 Link Layer

The SuperSpeed link layer has mechanisms that ensure a bit error rate less than 1 bit in 10^{20} bits for header packets. The link layer uses a number of techniques including packet framing ordered sets, link level flow control and retries to ensure reliable end-to-end delivery for header packets.

4.4.10.3 Protocol Layer

The SuperSpeed protocol layer depends on a 32-bit CRC appended to the Data Payload and a timeout coupled with retries to ensure that reliable data is provided to the application.

4.4.11 Efficiency

SuperSpeed efficiency is dependent on a number of factors including 8b/10b symbol encoding, packet structure and framing, link level flow control, and protocol overhead. At a 5 Gbps signaling rate with 8b/10b encoding, the raw throughput is 500 MBps. When link flow control, packet framing, and protocol overhead are considered, it is realistic for 400 MBps or more to be delivered to an application.

5 Mechanical

This chapter defines form, fit and function of the USB 3.0 connectors and cable assemblies. It contains the following:

- Connector mating interfaces
- Cables and cable assemblies
- Electrical requirements
- Mechanical and environmental requirements
- Implementation notes and guidelines

The intention of this chapter is to enable connector, system, and device designers and manufacturers to build, qualify, and use the USB 3.0 connectors, cables, and cable assemblies.

If any part of this chapter conflicts with the USB 2.0 specification, the USB 3.0 specification always supersedes the USB 2.0 specification.

5.1 Objective

The mechanical layer specification has been developed with the following objectives:

- Supporting 5 Gbps data rate
- Backward compatible with USB 2.0
- Minimizing connector form factor variations
- Managing EMI
- Supporting On-The-Go (OTG)
- Low cost

5.2 Significant Features

This section identifies the significant features of the USB 3.0 connectors and cable assemblies specification. The purpose of this section is not to present all the technical details associated with each major feature, but rather to highlight their existence. Where appropriate, this section references other parts of the document where further details can be found.

5.2.1 Connectors

The USB 3.0 specification defines the following connectors:

- USB 3.0 Standard-A plug and receptacle
- USB 3.0 Standard-B plug and receptacle
- USB 3.0 Powered-B plug and receptacle
- USB 3.0 Micro-B plug and receptacle
- USB 3.0 Micro-A plug
- USB 3.0 Micro-AB receptacle

Table 5-1 lists the compatible plugs and receptacles.

Table 5-1. Plugs Accepted By Receptacles

Receptacle	Plugs Accepted
USB 2.0 Standard-A	USB 2.0 Standard-A or USB 3.0 Standard-A
USB 3.0 Standard-A	USB 3.0 Standard-A or USB 2.0 Standard-A
USB 2.0 Standard-B	USB 2.0 Standard-B
USB 3.0 Standard-B	USB 3.0 Standard-B or USB 2.0 Standard-B
USB 3.0 Powered-B	USB 3.0 Powered-B, USB 3.0 Standard-B, or USB 2.0 Standard-B
USB 2.0 Micro-B	USB 2.0 Micro-B
USB 3.0 Micro-B	USB 3.0 Micro-B or USB 2.0 Micro-B
USB 2.0 Micro-AB	USB 2.0 Micro-B or USB 2.0 Micro-A
USB 3.0 Micro-AB	USB 3.0 Micro-B, USB 3.0 Micro-A, USB 2.0 Micro-B, or USB 2.0 Micro-A

5.2.1.1 USB 3.0 Standard-A Connector

The USB 3.0 Standard-A connector is defined as the host connector, supporting the SuperSpeed mode. It has the same mating interface as the USB 2.0 Standard-A connector, but with additional pins for two more differential pairs and a drain. Refer to Section 5.3.1.2 for pin assignments and descriptions.

A USB 3.0 Standard-A receptacle accepts either a USB 3.0 Standard-A plug or a USB 2.0 Standard-A plug. Similarly, a USB 3.0 Standard-A plug can be mated with either a USB 3.0 Standard-A receptacle or a USB 2.0 Standard-A receptacle.

A unique color coding is recommended for the USB 3.0 Standard-A connector plastic housings to help users distinguish the USB 3.0 Standard-A connector from the USB 2.0 Standard-A connector (refer to Section 5.3.1.3 for details).

5.2.1.2 USB 3.0 Standard-B Connector

The USB 3.0 Standard-B connector is defined for relatively large, stationary peripherals, such as external hard drives and printers. It is defined so that the USB 3.0 Standard-B receptacle accepts either a USB 3.0 Standard-B plug or a USB 2.0 Standard-B plug. Inserting a USB 3.0 Standard-B plug into a USB 2.0 Standard-B receptacle is physically disallowed (refer to Section 5.3.2 for details).

5.2.1.3 USB 3.0 Powered-B Connector

The USB 3.0 Powered-B connector is defined to allow a USB 3.0 device to provide power to a USB adaptor without the need for an external power supply. It is identical to the USB 3.0 Standard-B connector in form factor, but has two more pins- one for power (DPWR) and one for ground (DGND). See Section 5.3.3 for details.

5.2.1.4 USB 3.0 Micro-B Connector

The USB 3.0 Micro-B connector is defined for small handheld devices. It is compatible with the USB 2.0 Micro-B connector; i.e., a USB 2.0 Micro-B plug works in a USB 3.0 Micro-B receptacle. Section 5.3.4 defines the USB 3.0 Micro connector family.

5.2.1.5 USB 3.0 Micro-AB and USB 3.0 Micro-A Connectors

The USB 3.0 Micro-AB receptacle is similar to the USB 3.0 Micro-B receptacle, except for different keying. It accepts a USB 3.0 Micro-A plug, a USB 3.0 Micro-B plug, a USB 2.0 Micro-A plug, or a USB 2.0 Micro-B plug. The USB 3.0 Micro-AB receptacle is only allowed on OTG products, which may function as either a host or device. All other uses of the USB 3.0 Micro-AB receptacle are prohibited.

The USB 3.0 Micro-A plug is similar to the USB 3.0 Micro-B plug, except for different keying and ID pin connections. The USB 3.0 Micro-A plug, the USB 3.0 Micro-AB receptacle, and the USB 3.0 Micro-B receptacle and plug all belong to the USB 3.0 Micro connector family since their interfaces differ only in keying. Similar to the USB 2.0 Micro-A plug, the USB 3.0 Micro-A plug is defined for OTG applications only.

5.2.2 Compliant Cable Assemblies

The USB 3.0 specification defines the following cable assemblies:

- USB 3.0 Standard-A plug to USB 3.0 Standard-B plug
- USB 3.0 Standard-A plug to USB 3.0 Micro-B plug
- USB 3.0 Standard-A plug to USB 3.0 Standard-A plug
- USB 3.0 Micro-A plug to USB 3.0 Micro-B plug
- USB 3.0 Micro-A plug to USB 3.0 Standard-B plug
- Captive cable with USB 3.0 Standard-A plug
- Permanently attached cable with USB 3.0 Micro-A plug
- Permanently attached cable with USB 3.0 Powered-B plug

A captive cable is a cable assembly that has a Standard-A plug on one end and that is either permanently attached or has a vendor-specific connector on the other end. A permanently attached cable is directly wired to the device and it is not detachable from the device. This specification does not define how the vendor-specific connector or permanent attachment shall be done on the device side.

For electrical compliance purpose, a USB 3.0 captive cable (permanently attached or with vendor-specific connector on the device end) shall be considered part of the USB 3.0 device.

No other types of cable assemblies are allowed by this specification. Section 5.5 provides detailed discussion on USB 3.0 cable assemblies.

5.2.3 Raw Cables

Due to EMI and signal integrity requirements, each cable differential pair used for the SuperSpeed lines in a USB 3.0 cable assembly must be shielded; the Unshielded Twisted Pair (UTP) used for USB 2.0 is not allowed for SuperSpeed. Section 5.4 defines the cable construction for USB 3.0.

5.3 Connector Mating Interfaces

This section defines the connector mating interfaces, including the connector interface drawings, pin assignments and descriptions.

5.3.1 USB 3.0 Standard-A Connector

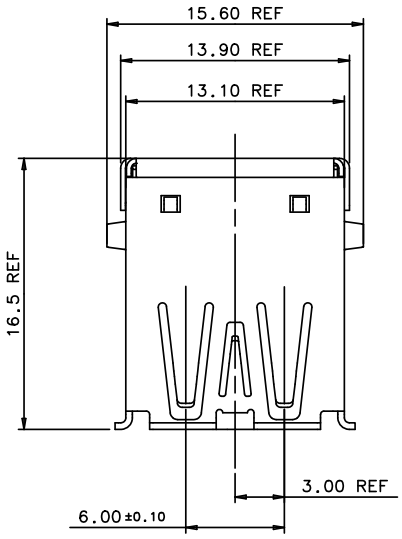
5.3.1.1 Interface Definition

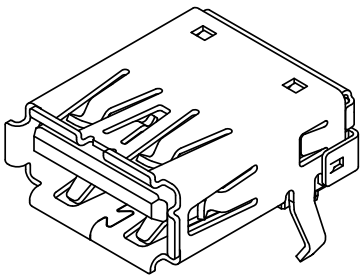
Figure 5-1 to Figure 5-4 show, respectively, the USB 3.0 Standard-A receptacle and plug interface dimensions, as well as the reference footprints for the USB 3.0 Standard-A receptacle. Note that only the dimensions that govern the mating interoperability are specified. All the REF dimensions are provided for reference only, not hard requirements.

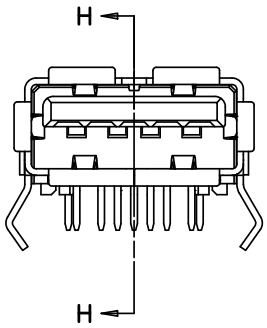
Although the USB 3.0 Standard-A connector has basically the same form factor as the USB 2.0 Standard-A connector, it has significant differences inside. Below are the key features and design areas that need attention:

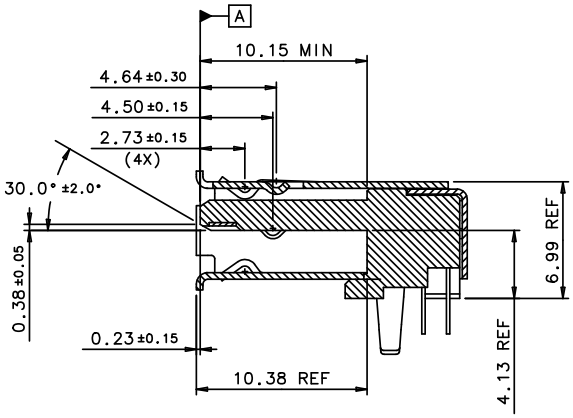
- Besides the VBUS, D-, D+, and GND pins that are required for USB 2.0, the USB 3.0 Standard-A connector includes five more pins–two differential pairs plus one ground (GND_DRAIN). The two added differential pairs are for SuperSpeed data transfer, supporting dual simplex SuperSpeed signaling; the added GND_DRAIN pin is for drain wire termination, managing signal integrity, and EMI performance.
- The contact areas of the five SuperSpeed pins are located towards the front of the receptacle as the blades, while the four USB 2.0 pins towards the back of the receptacle as the beams or springs. Accordingly in the plug, the SuperSpeed contacts, as the beams, seat behind the USB 2.0 blades. In other words, the USB 3.0 Standard-A connector has a two-tier contact system.
- The tiered-contact approach within the Standard-A connector form factor inevitably results in less contact area to work with, as compared to the USB 2.0 Standard-A connector. The connector interface dimensions take into consideration of contact mating requirements between the USB 3.0 Standard-A receptacle and USB 3.0 Standard-A plug, the USB 3.0 Standard-A receptacle and USB 2.0 Standard-A plug, and the USB 2.0 Standard-A receptacle and USB 3.0 Standard-A plug. Connector designers should carefully consider those aspects in design details.
- The connector interface definition comprehends the need to avoid shorting between the SuperSpeed and USB 2.0 pins during insertion when plugging a USB 2.0 Standard-A plug into a USB 3.0 Standard-A receptacle, or a USB 3.0 Standard-A plug into a USB 2.0 Standard-A receptacle. Connector designers should be conscious of this when detailing out designs.
- There may be some increase in the USB 3.0 Standard-A receptacle connector depth (into a system board) to support the two-tiered-contacts, as compared to the USB 2.0 Standard-A receptacle.
- The through-hole footprints in Figure 5-3 and Figure 5-4 are shown as examples. Other footprints, such as SMT (surface mount) are also allowed.

- Drawings for stacked USB 3.0 Standard-A receptacles are not shown in this specification. But they are allowed, as long as they meet all the electrical and mechanical requirements defined in this specification. In fact, a double-stacked USB 3.0 Standard-A receptacle is expected to be a common application just like the double-stacked USB 2.0 Standard-A receptacle that has been widely used in PCs. When designing a stacked USB 3.0 Standard-A receptacle, efforts must be made to minimize impedance discontinuity of the top connector in the stack because of its long electrical length. Figure 5-4 shows an example or reference footprint for a double-stacked Standard-A receptacle connector. Note that pins 1 to 9 correspond to the lower port, while pins 10 to 18 correspond to the upper port. Section 5.8.3 offers further discussion on the stacked connector.
- Attention must be paid to the high speed electrical design of USB 3.0 Standard-A connectors. Besides minimizing the connector impedance discontinuities, crosstalk among the SuperSpeed pairs and USB 2.0 D+/D- pair should also be minimized.





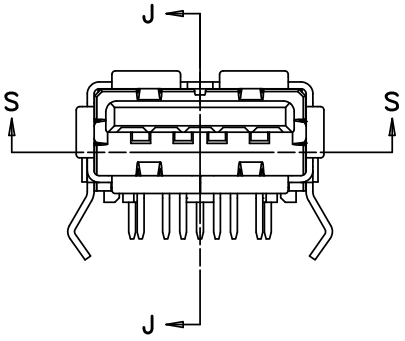


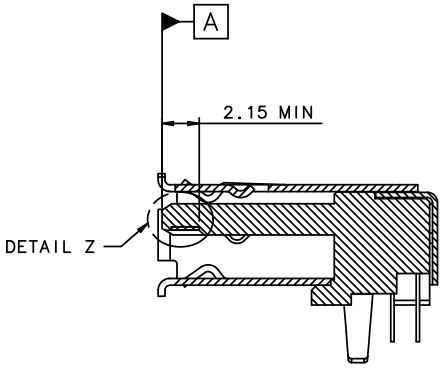


SECTION H-H

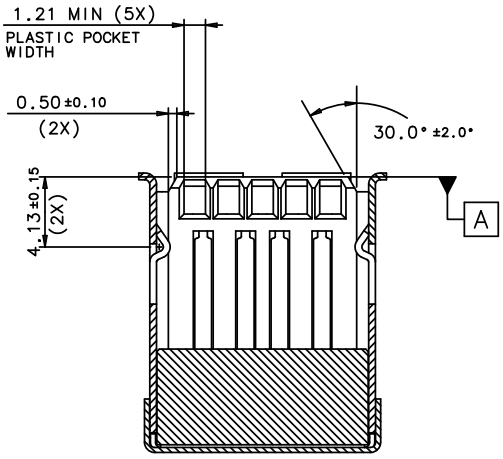
NOTES:
1) NON-DIMENSIONED GEOMETRY FOR REFERENCE ONLY, SUBJECT TO CHANGE.
2) DRAWING FOR MATING INTERFACE DIMENIONS ONLY. VIEWS MAY NOT SHOW REALISTIC MANUFACTURING CONDITION.

continued

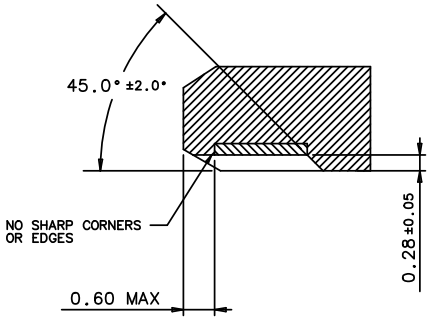




SECTION J-J



SECTION S-S



DETAIL Z

continued

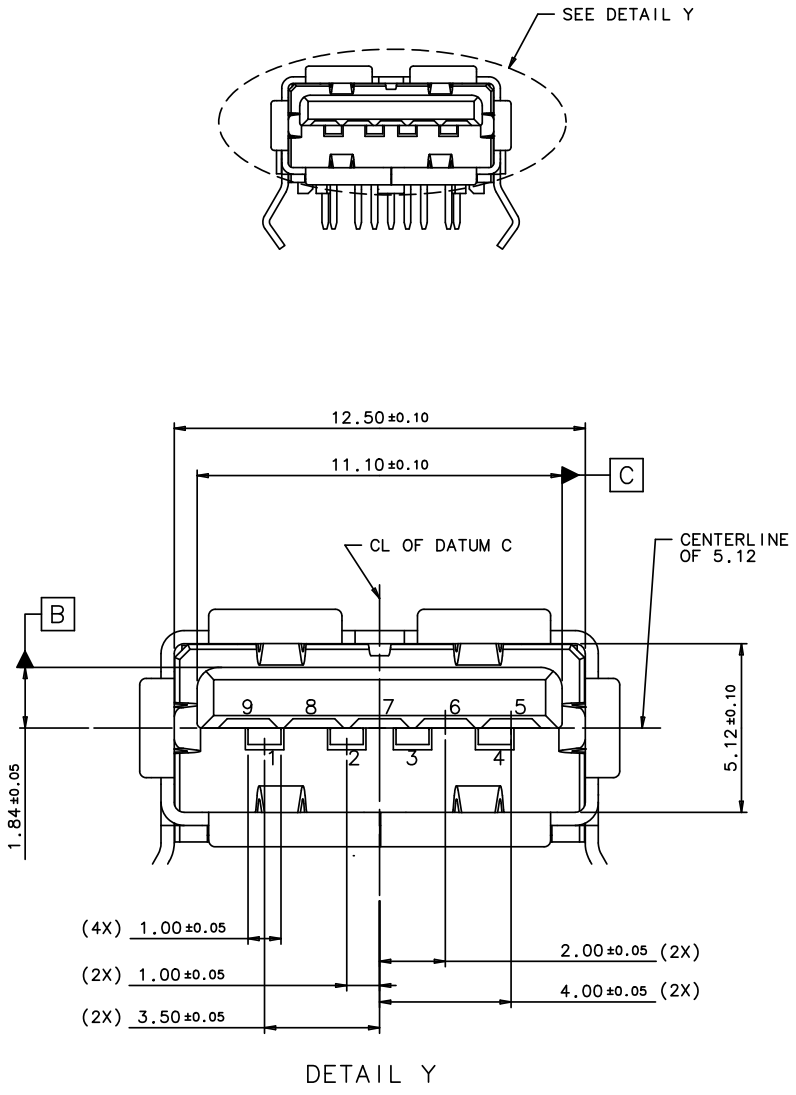
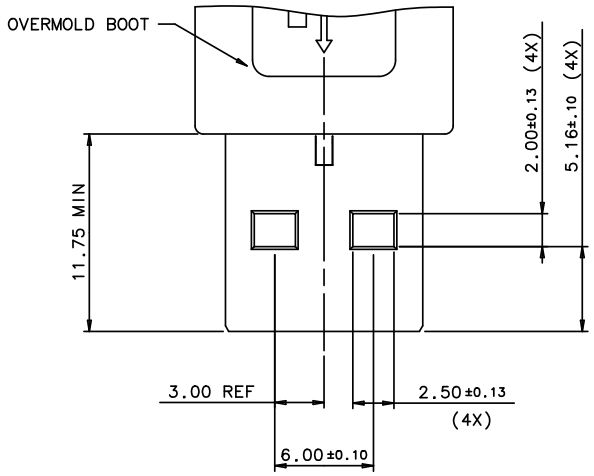
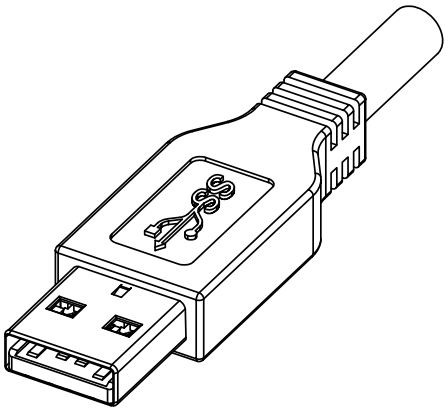
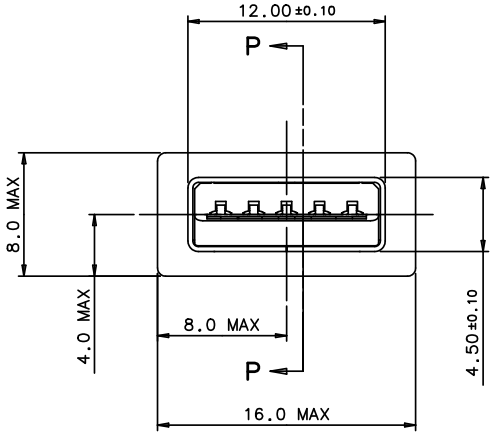
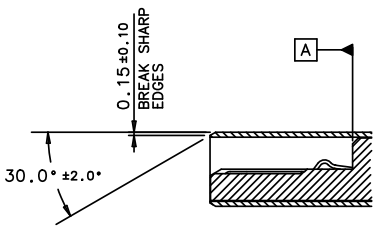


Figure 5-1. USB 3.0 Standard-A Receptacle Interface Dimensions





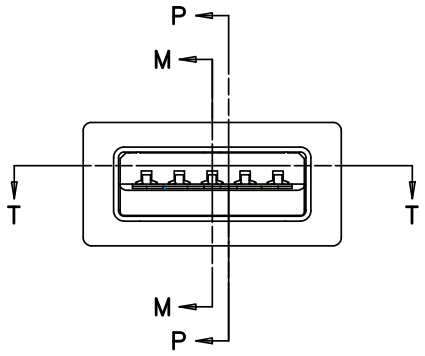


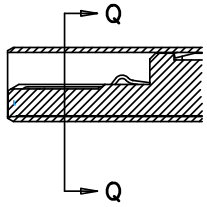


SECTION P-P

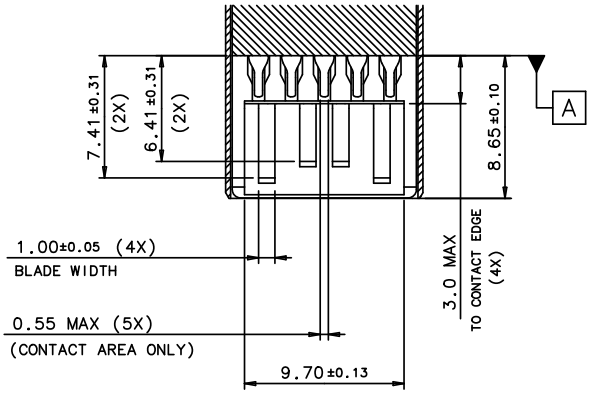
NOTES:
1) NON-DIMENSIONED GEOMETRY FOR REFERENCE ONLY, SUBJECT TO CHANGE
2) DRAWING FOR MATING INTERFACE DIMENSIONS ONLY. VIEWS MAY NOT SHOW REALISTIC MANUFACTURING CONDITION.

continued

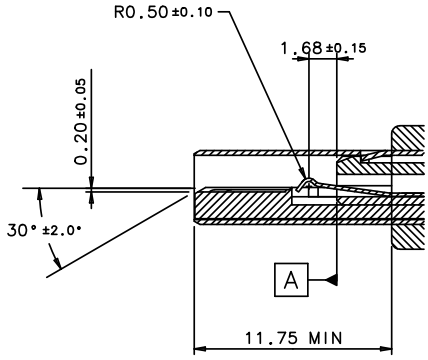




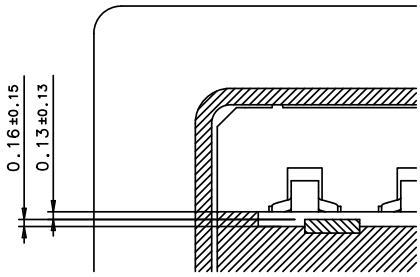
SECTION P-P



SECTION T-T



SECTION M-M



SECTION Q-Q

continued

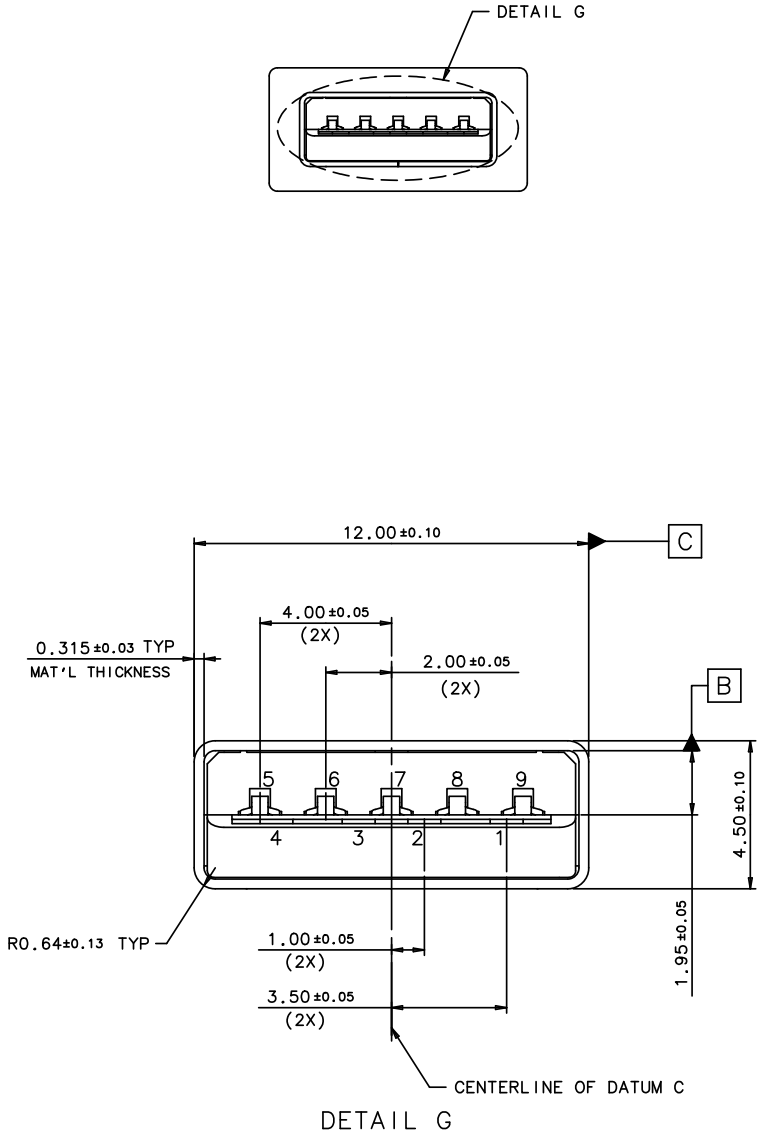


Figure 5-2. USB 3.0 Standard-A Plug Interface Dimensions

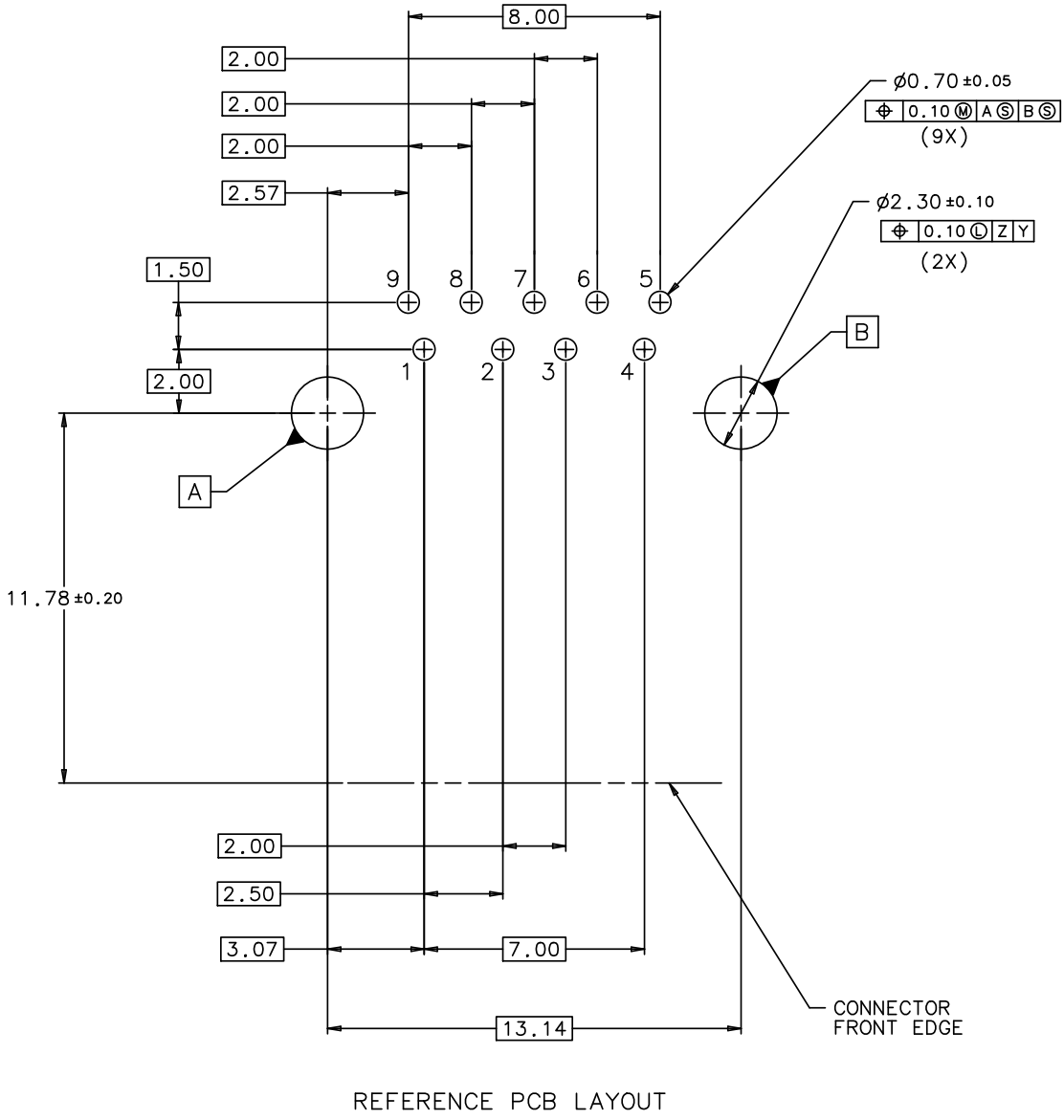


Figure 5-3. Reference Footprint for the USB 3.0 Standard-A Receptacle

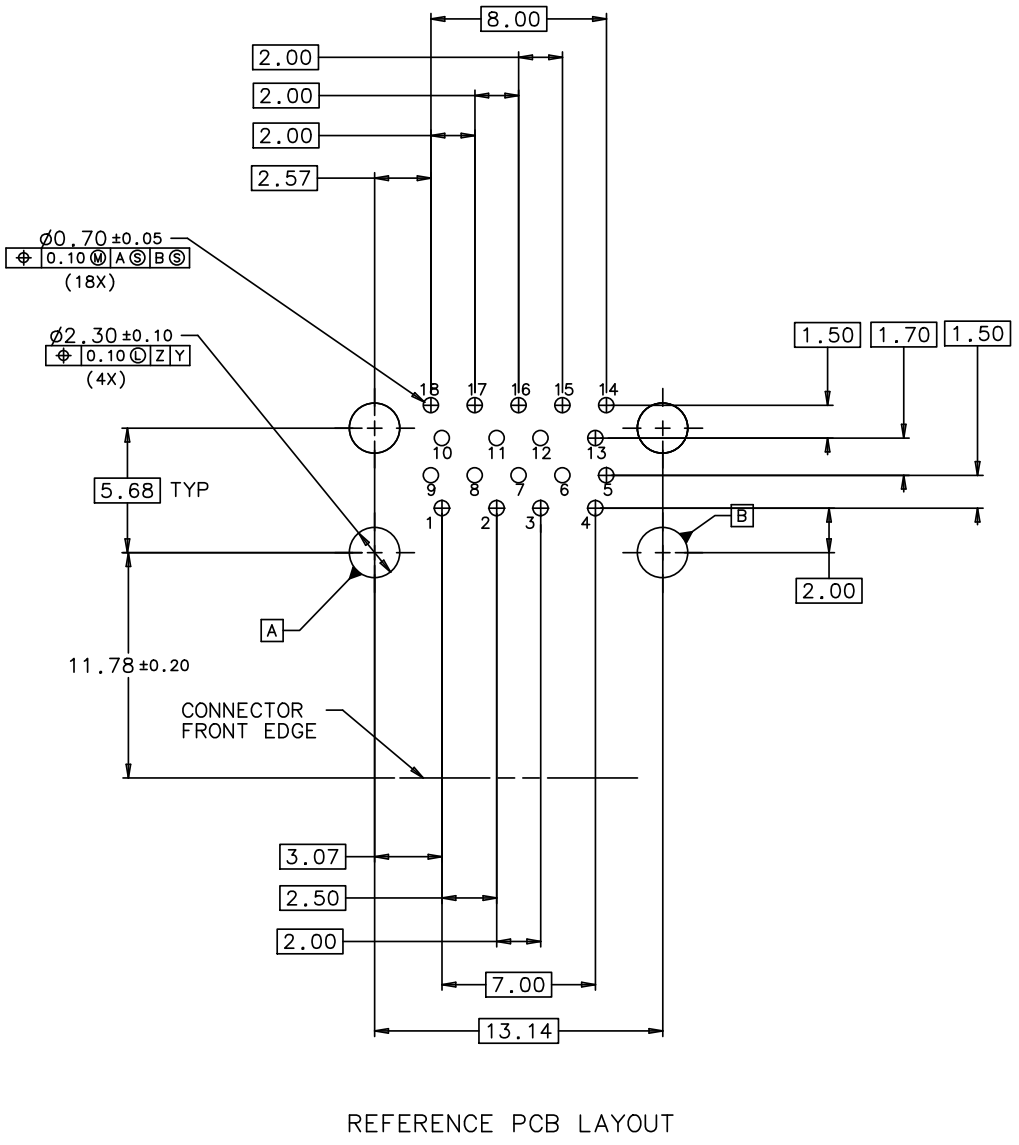


Figure 5-4. Reference Footprint for the USB 3.0 Double-Stacked Standard-A Receptacle

5.3.1.2 Pin Assignments and Description

The usage and assignments of the nine pins in the USB 3.0 Standard-A connector are defined in Table 5-2.

Table 5-2. USB 3.0 Standard-A Connector Pin Assignments

Pin Number	Signal Name	Description	Mating Sequence
1	VBUS	Power	Second
2	D-	USB 2.0 differential pair	Third
3	D+		
4	GND	Ground for power return	Second
5	StdA_SSRX-	SuperSpeed receiver differential pair	Last
6	StdA_SSRX+		
7	GND_DRAIN	Ground for signal return	
8	StdA_SSTX-	SuperSpeed transmitter differential pair	
9	StdA_SSTX+		
Shell	Shield	Connector metal shell	First

Note: Tx and Rx are defined from the host perspective

The physical location of the pins in the connector is illustrated in Figure 5-1 to Figure 5-4. Note that pins 1 to 4 are referred to as the USB 2.0 pins, while pins 5 to 9 are referred to as the SuperSpeed pins.

5.3.1.3 USB 3.0 Standard-A Connector Color Coding

Since both the USB 2.0 Standard-A and USB 3.0 Standard-A receptacles may co-exist on a host, color coding is recommended for the USB 3.0 Standard-A connector (receptacle and plug) housings to help users distinguish it from the USB 2.0 Standard-A connector.

Blue (Pantone 300C) is the recommended color for the USB 3.0 Standard-A receptacle and plug plastic housings. When the recommended color is used, connector manufacturers and system integrators should make sure that the blue-colored receptacle housing is visible to users. Figure 5-5 illustrates the color coding recommendation for the USB 3.0 Standard-A connector.

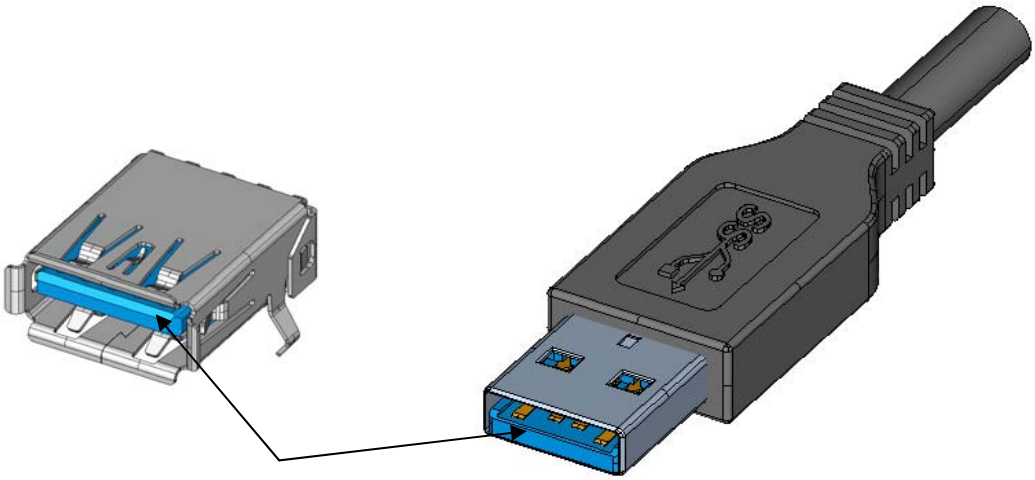
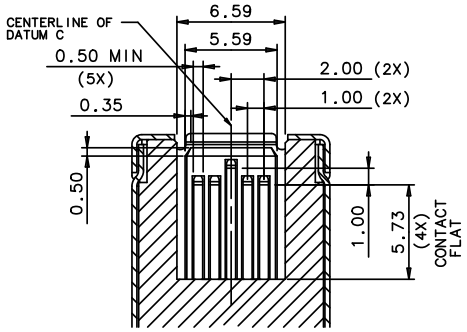


Figure 5-5. Illustration of Color Coding Recommendation for USB 3.0 Standard-A Connector

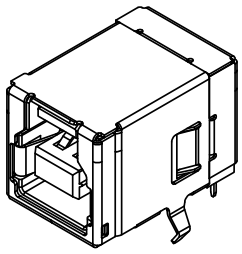
5.3.2 USB 3.0 Standard-B Connector

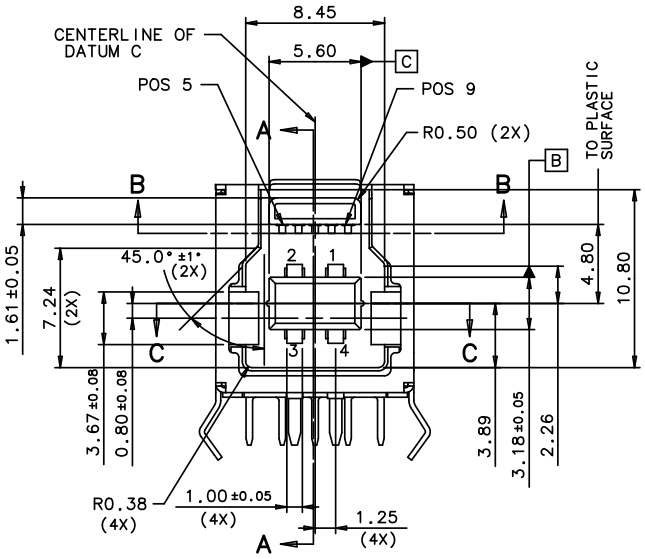
5.3.2.1 Interface Definition

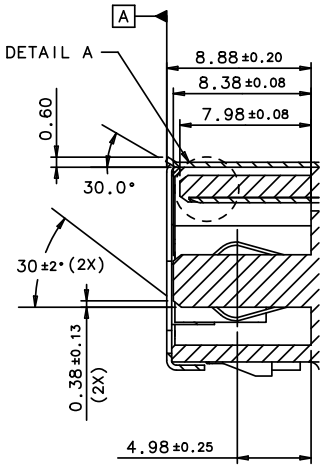
Figure 5-6 to Figure 5-8 show, respectively, the USB 3.0 Standard-B receptacle and plug interface dimensions, as well as the reference footprint.



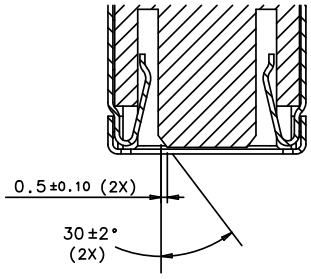
SECTION B-B



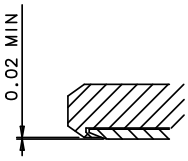




SECTION A-A

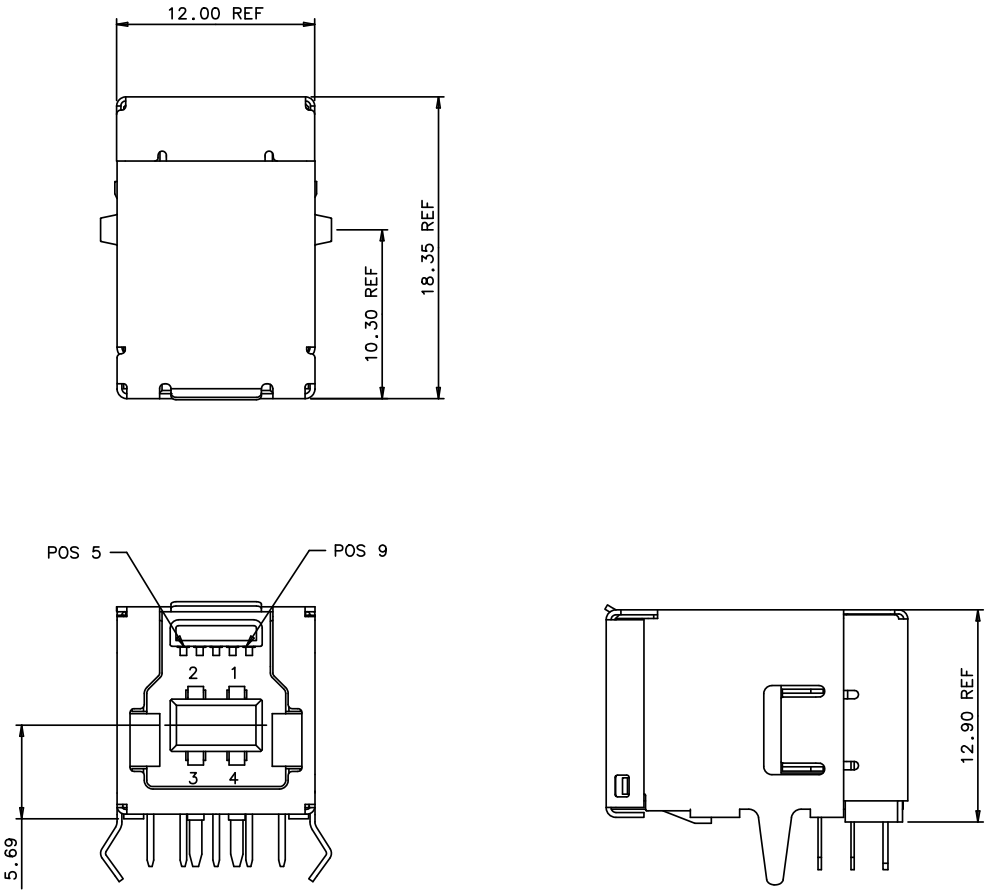


SECTION C-C



DETAIL A

continued



NOTES:
1) CRITICAL DIMENSIONS ARE TOLERANCED AND SHALL NOT BE DEVIATED.
2) GENERAL TOLERANCE IS ±0.10 OTHERWISE THE SPECIFIED TOLERANCE APPLIES.
3) ALL DIMENSIONS ARE IN MILLIMETERS.
4) DIMENSIONS THAT ARE LABELED REF ARE TYPICAL DIMENSIONS AND MAY VARY FROM MANUFACTURER TO MANUFACTURER.
5) NON-DIMENSIONED GEOMETRY FOR REFERENCE ONLY, SUBJECT TO CHANGE.

Figure 5-6. USB 3.0 Standard-B Receptacle Interface Dimensions

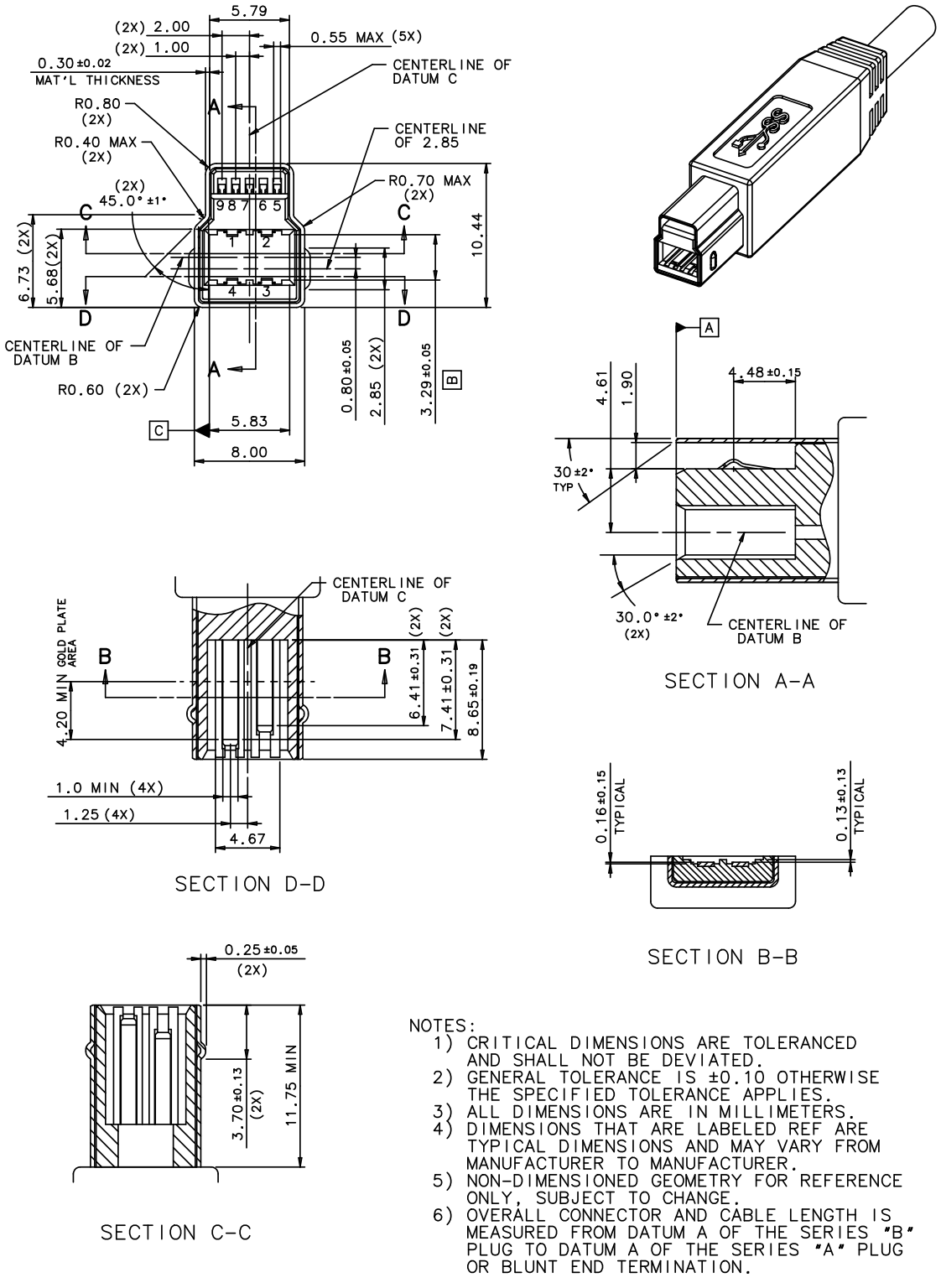


Figure 5-7. USB 3.0 Standard-B Plug Interface Dimensions

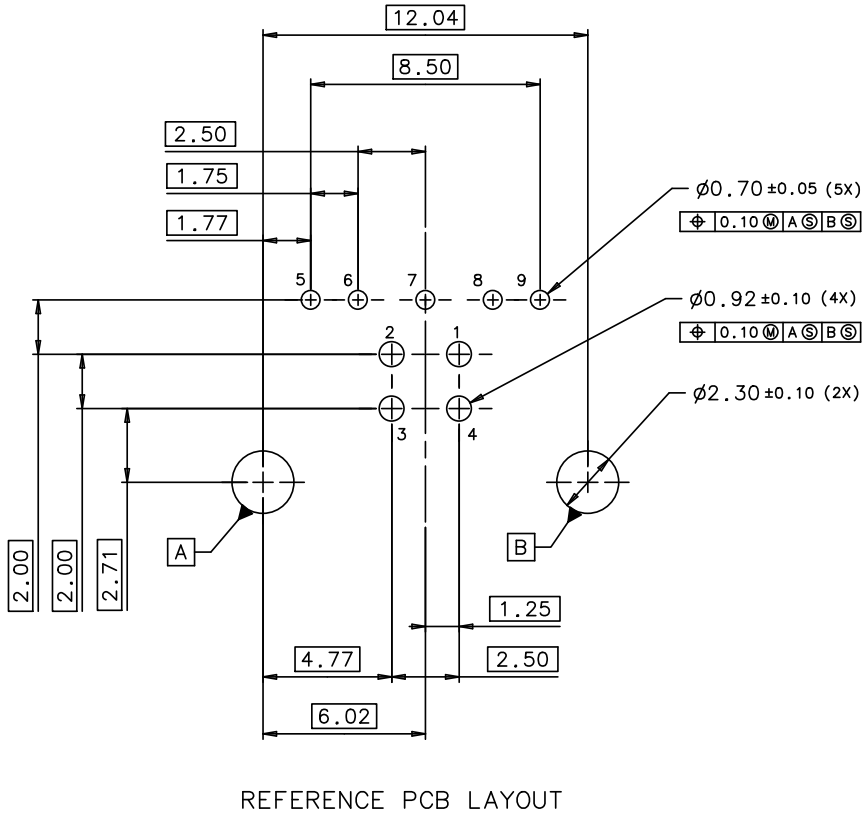


Figure 5-8. Reference Footprint for the USB 3.0 Standard-B Receptacle

The USB 3.0 Standard-B receptacle interfaces have two portions: the USB 2.0 interface and the SuperSpeed interface. The USB 2.0 interface consists of pins 1 to 4, while the SuperSpeed interface consists of pins 5 to 9.

When a USB 2.0 Standard-B plug is inserted into the USB 3.0 Standard-B receptacle, only the USB 2.0 interface is engaged, and the link will not take advantage of the SuperSpeed capability. However, since the USB 3.0 SuperSpeed portion is visibly not mated when a USB 2.0 Standard-B plug is inserted in the USB 3.0 Standard-B receptacle, users will get the visual feedback that the cable plug is not matched with the receptacle. Only when a USB 3.0 Standard-B plug is inserted into the USB 3.0 Standard-B receptacle, is the interface completely visibly engaged.

5.3.2.2 Pin Assignments and Description

The usage and assignments of the nine pins in the USB 3.0 Standard-B connector are defined in Table 5-3.

Table 5-3. USB 3.0 Standard-B Connector Pin Assignments

Pin Number	Signal Name	Description	Mating Sequence
1	VBUS	Power	Second
2	D-	USB 2.0 differential pair	Third or beyond
3	D+		
4	GND	Ground for power return	Second
5	StdB_SSTX-	SuperSpeed transmitter differential pair	Third or beyond
6	StdB_SSTX+		
7	GND_DRAIN	Ground for signal return	
8	StdB_SSRX-	SuperSpeed receiver differential pair	
9	StdB_SSRX+		
Shell	Shield	Connector metal shell	First

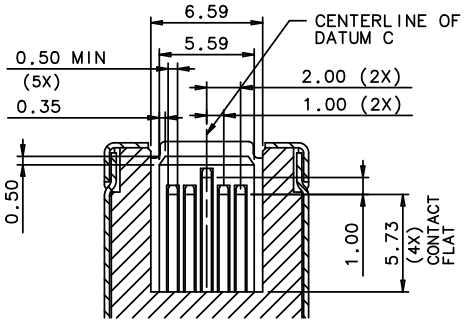
Note: Tx and Rx are defined from the device perspective

The physical location of the pins in the connector is illustrated in Figure 5-6 to Figure 5-8.

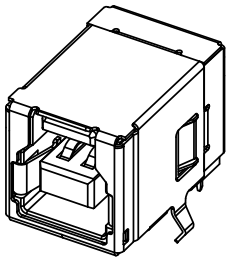
5.3.3 USB 3.0 Powered-B Connector

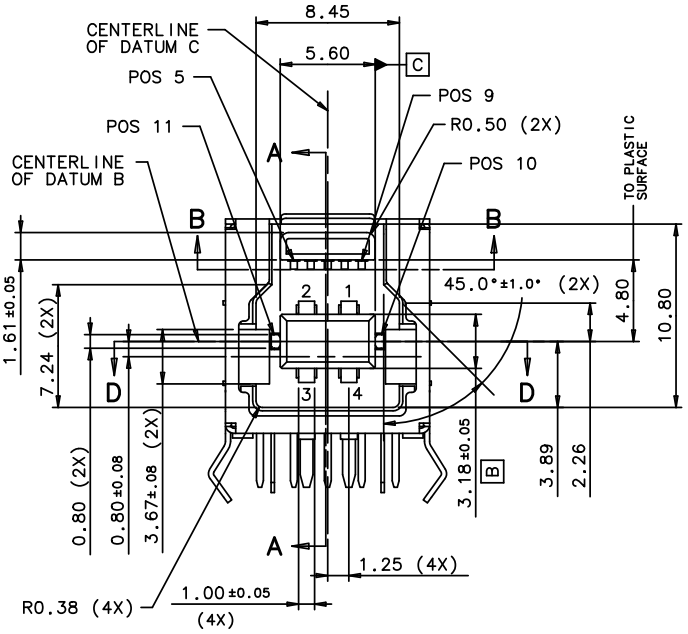
5.3.3.1 Interface Definition

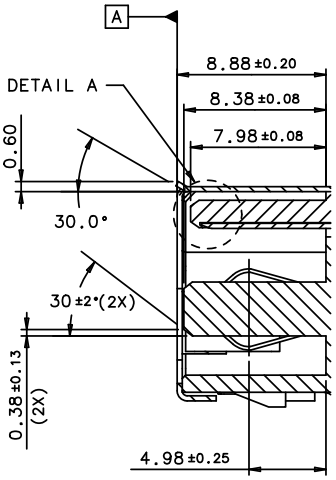
Figure 5-9 to Figure 5-11 show the USB 3.0 Powered-B receptacle and plug interface dimensions, as well as the reference footprint.



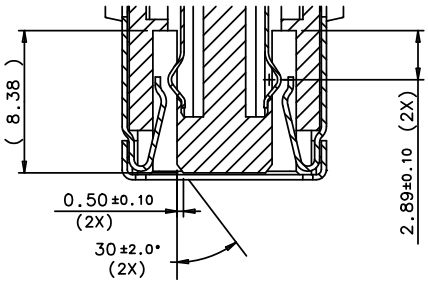
SECTION B-B



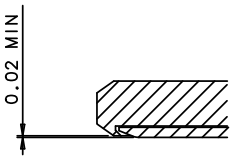




SECTION A-A

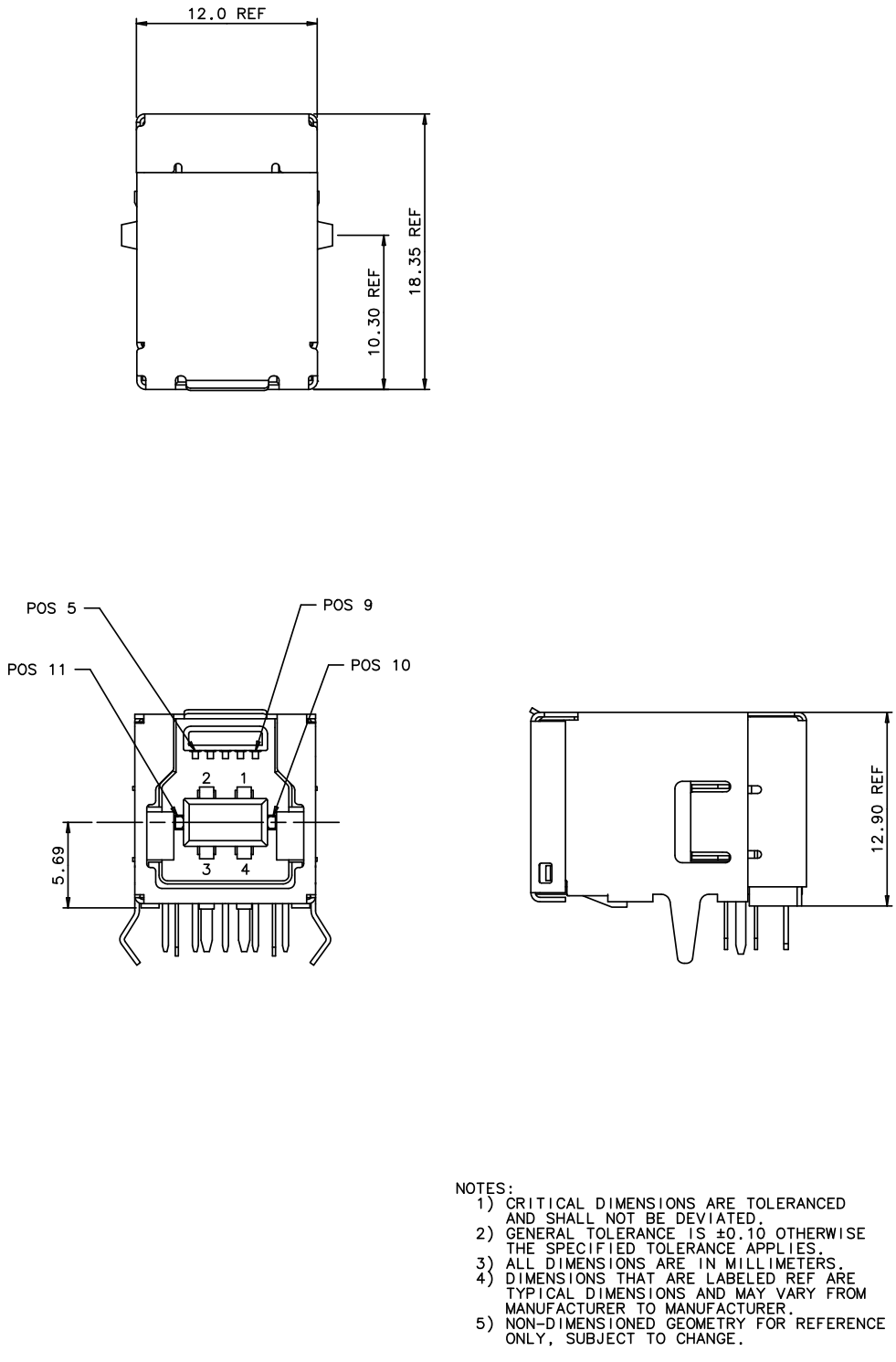


SECTION D-D



DETAIL A

continued



NOTES:
1) CRITICAL DIMENSIONS ARE TOLERANCED AND SHALL NOT BE DEVIATED.
2) GENERAL TOLERANCE IS ±0.10 OTHERWISE THE SPECIFIED TOLERANCE APPLIES.
3) ALL DIMENSIONS ARE IN MILLIMETERS.
4) DIMENSIONS THAT ARE LABELED REF ARE TYPICAL DIMENSIONS AND MAY VARY FROM MANUFACTURER TO MANUFACTURER.
5) NON-DIMENSIONED GEOMETRY FOR REFERENCE ONLY, SUBJECT TO CHANGE.

Figure 5-9. USB 3.0 Powered-B Receptacle Interface Dimensions

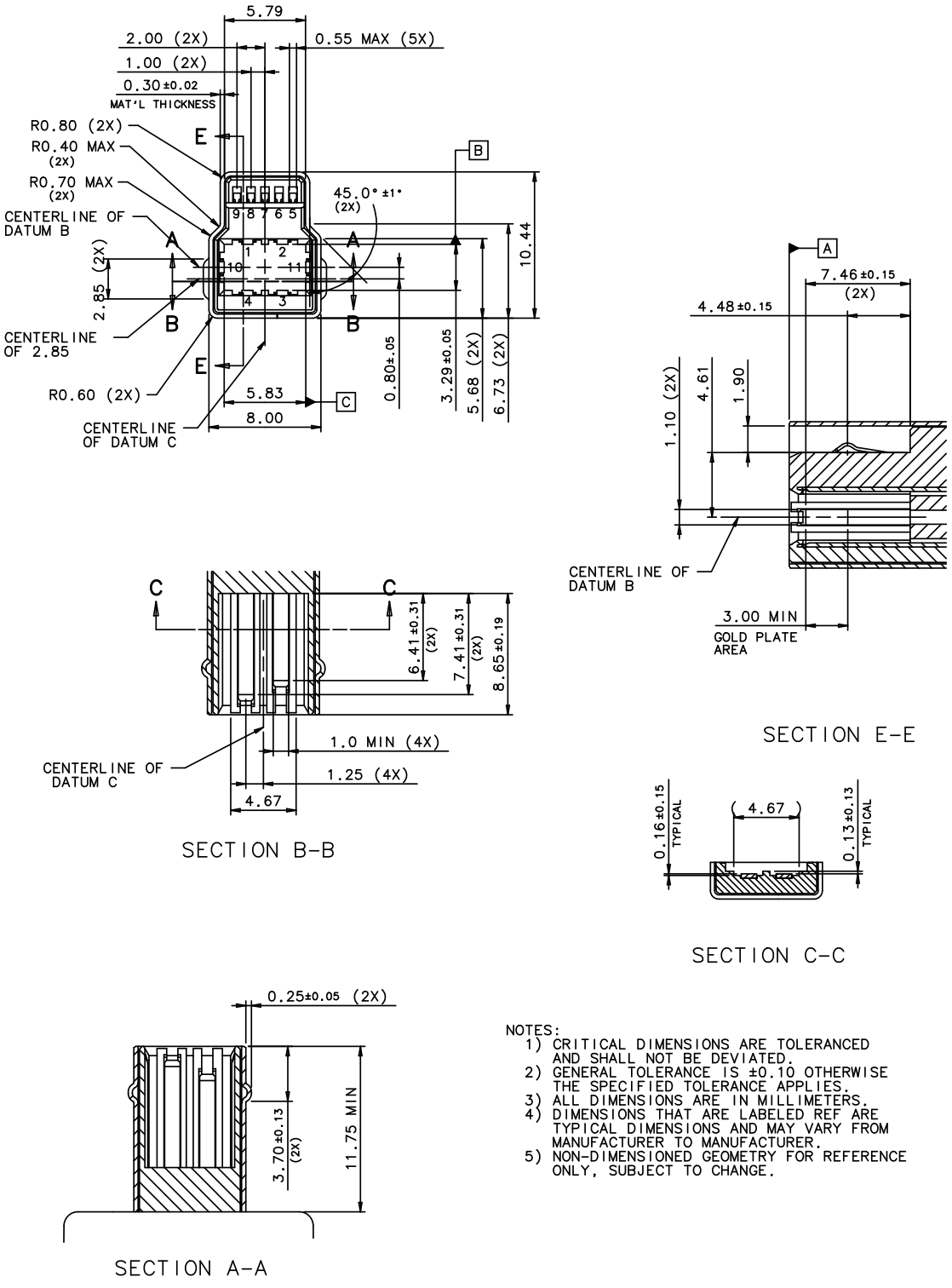


Figure 5-10. USB 3.0 Powered-B Plug Interface Dimensions

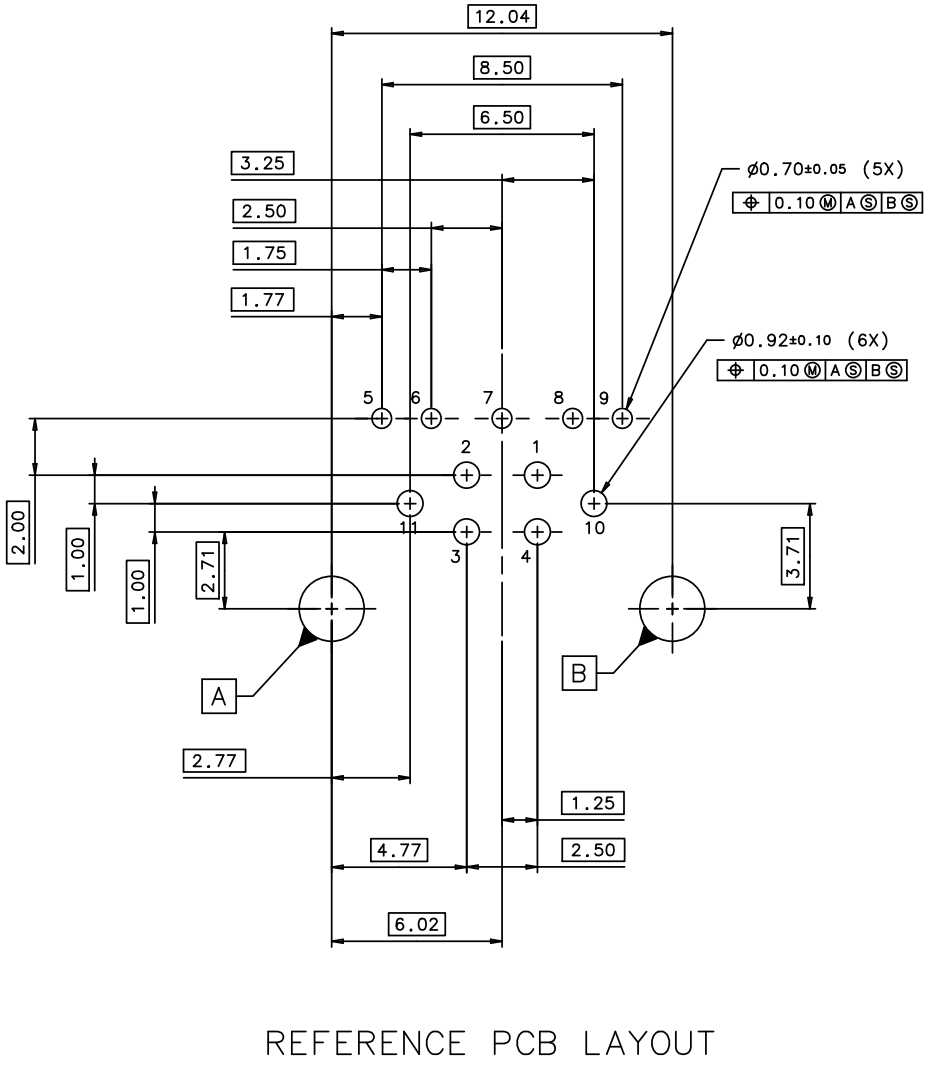


Figure 5-11. Reference Footprint for USB 3.0 Powered-B Receptacle

5.3.3.2 Pin Assignments and Descriptions

The usage and assignments of the 11 pins in the USB 3.0 Powered-B connector are defined in Table 5-4.

Table 5-4. USB 3.0 Powered-B Connector Pin Assignments

Pin Number	Signal Name	Description	Mating Sequence
1	VBUS	Power	Second
2	D-	USB 2.0 differential pair	Third or beyond
3	D+		
4	GND	Ground for power return	Second
5	StdB_SSTX-	SuperSpeed transmitter differential pair	Third or beyond
6	StdB_SSTX+		
7	GND_DRAIN	Ground for signal return	
8	StdB_SSRX-	SuperSpeed receiver differential pair	
9	StdB_SSRX+		
10	DPWR	Power provided by device	
11	DGND	Ground Return for DPWR	
Shell	Shield	Connector metal shell	First

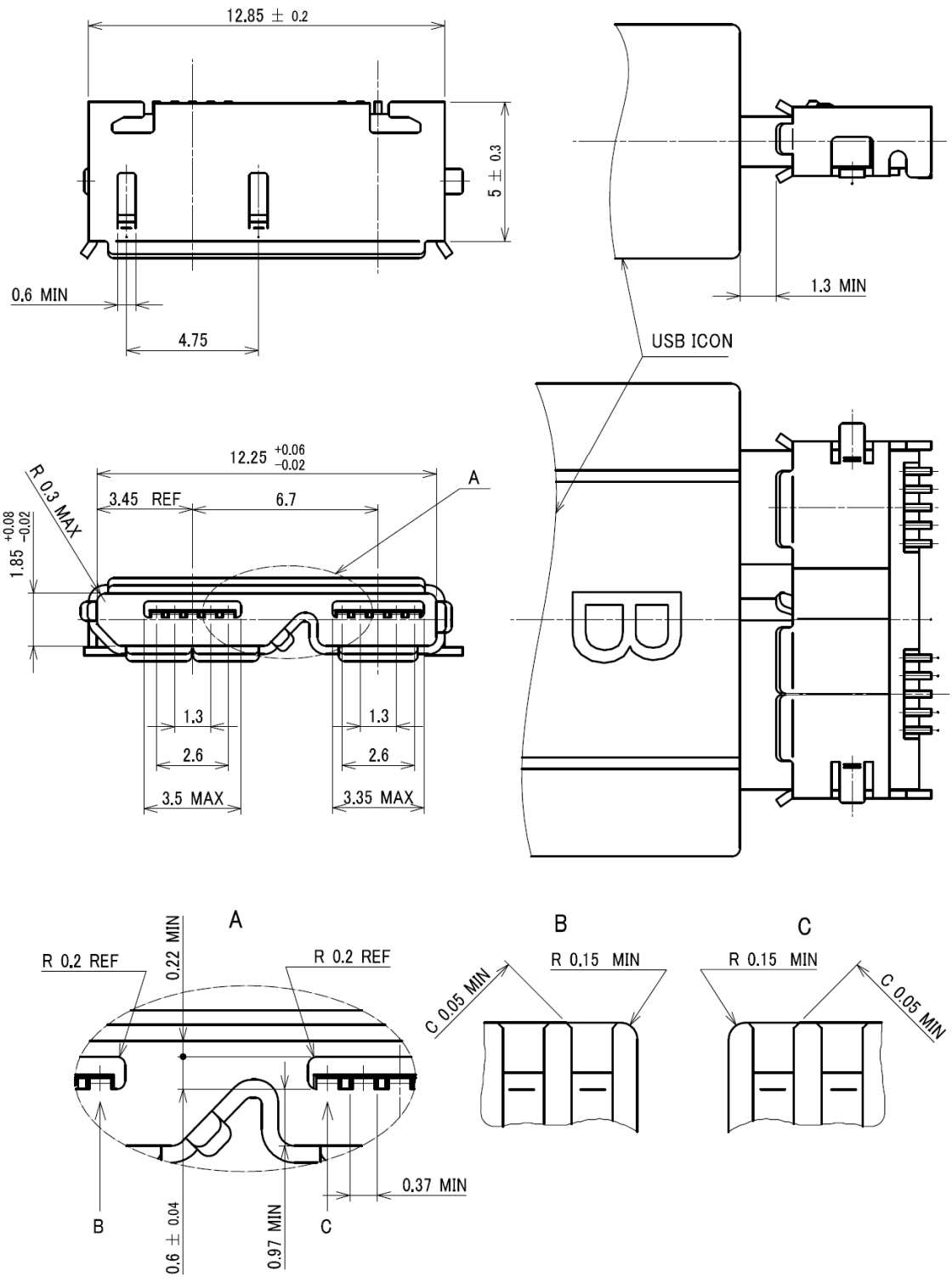
Note: Tx and Rx are defined from the device perspective

The physical location of the pins in the connector is illustrated in Figure 5-9 to Figure 5-11.

5.3.4 USB 3.0 Micro Connector Family

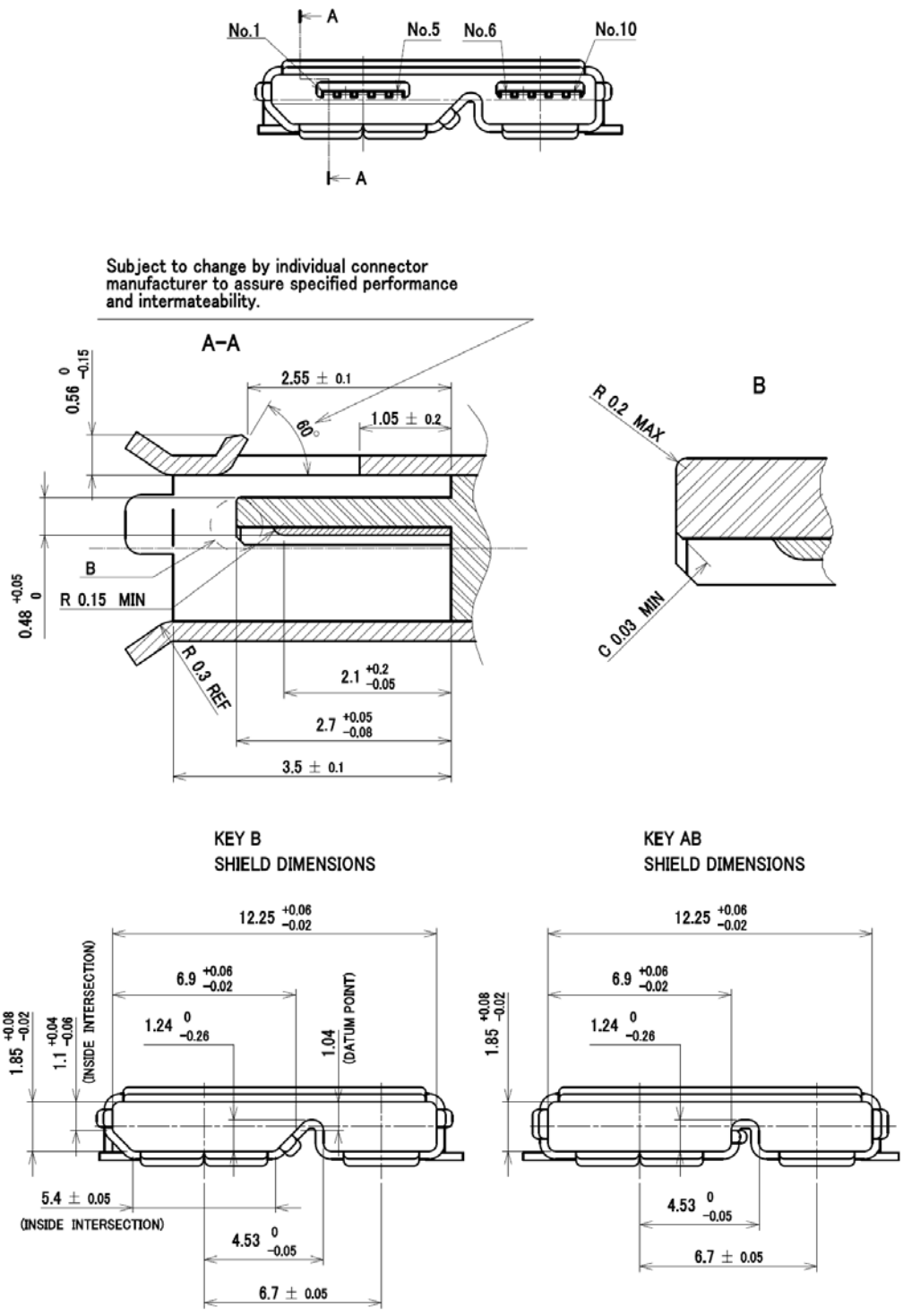
5.3.4.1 Interfaces Definition

The USB 3.0 Micro connector family consists of the USB 3.0 Micro-B receptacle, USB 3.0 Micro-AB receptacle, USB 3.0 Micro-B plug and USB 3.0 Micro-A plug. Figure 5-12 and Figure 5-13 show the USB 3.0 Micro family receptacle and plug interface dimensions. Note that only the dimensions that govern the mating interoperability are specified.



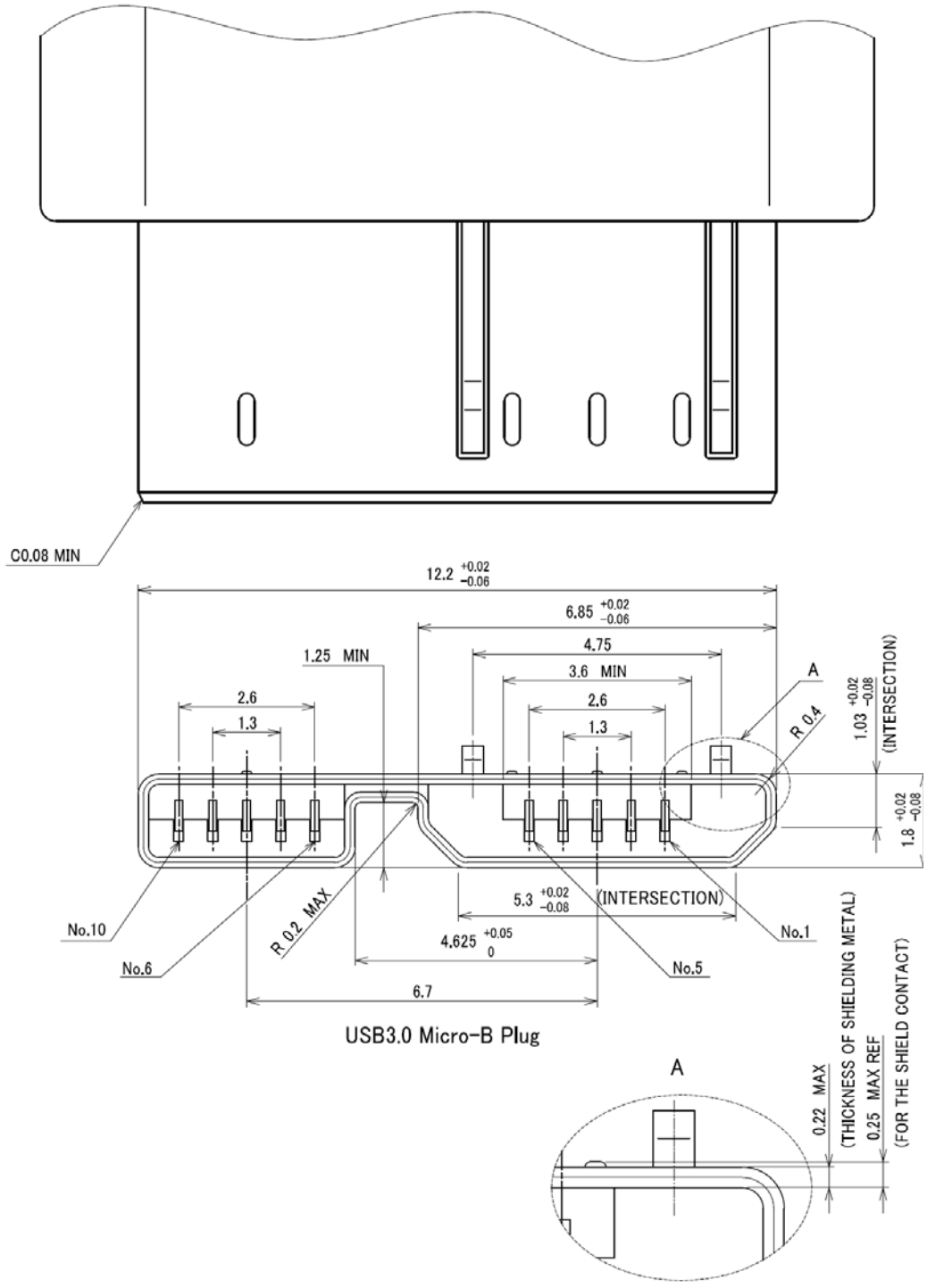
NOTE : General tolerance is +/− 0.05 mm, otherwise the specified tolerances apply.

continued



NOTE : Chamfer metals are optional with no sharp edges.

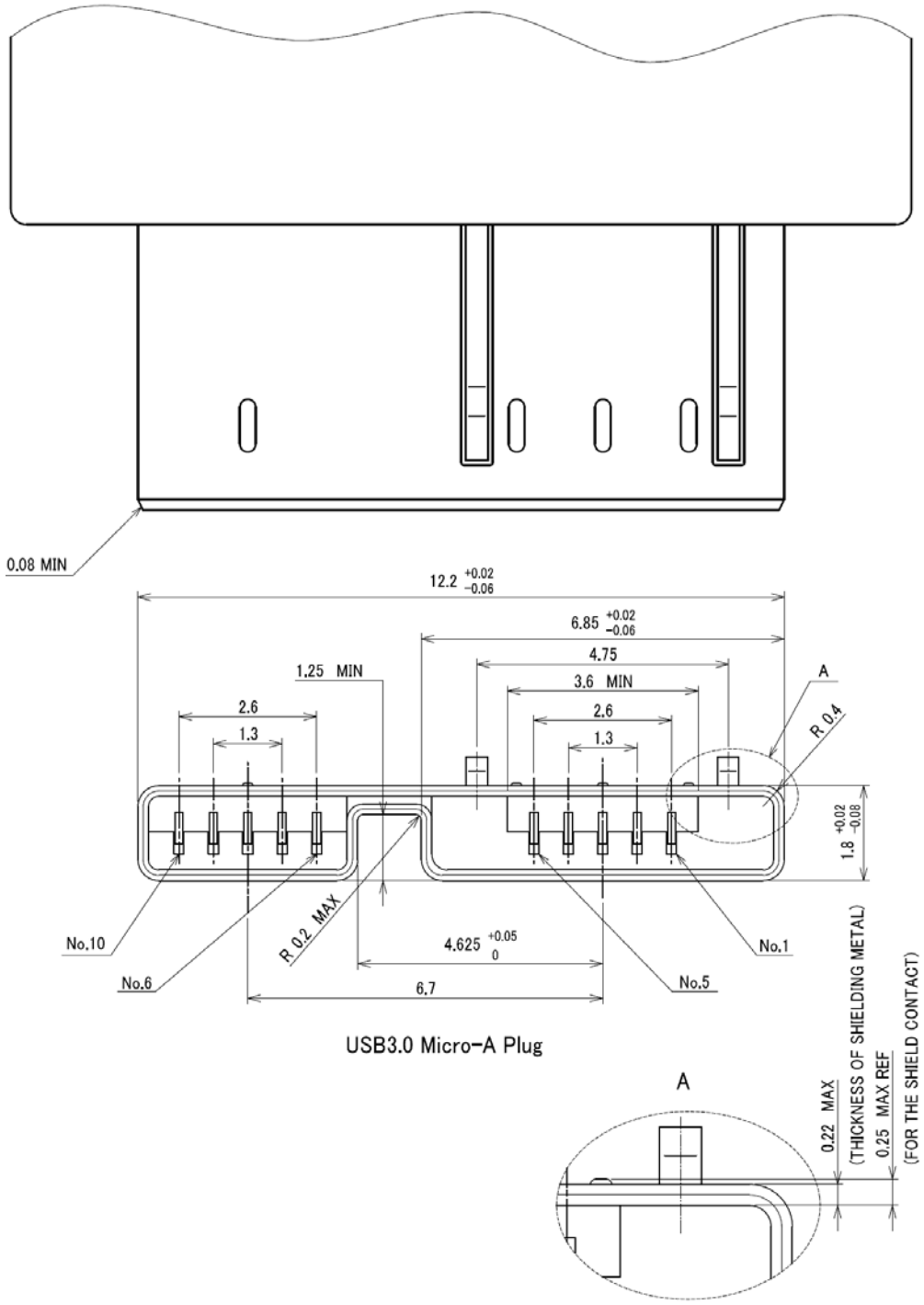
Figure 5-12. USB 3.0 Micro-B and -AB Receptacles Interface Dimensions



NOTES:

1. Dimensions that are labeled REF may vary from manufacturer to manufacturer.
2. General tolerance is +/−0.05mm, otherwise the specified tolerances apply.

continued



USB3.0 Micro-A Plug

NOTES:

1. Dimensions that are labeled REF may vary from manufacturer to manufacturer.
2. General tolerance is +/−0.05mm, otherwise the specified tolerances apply.

continued

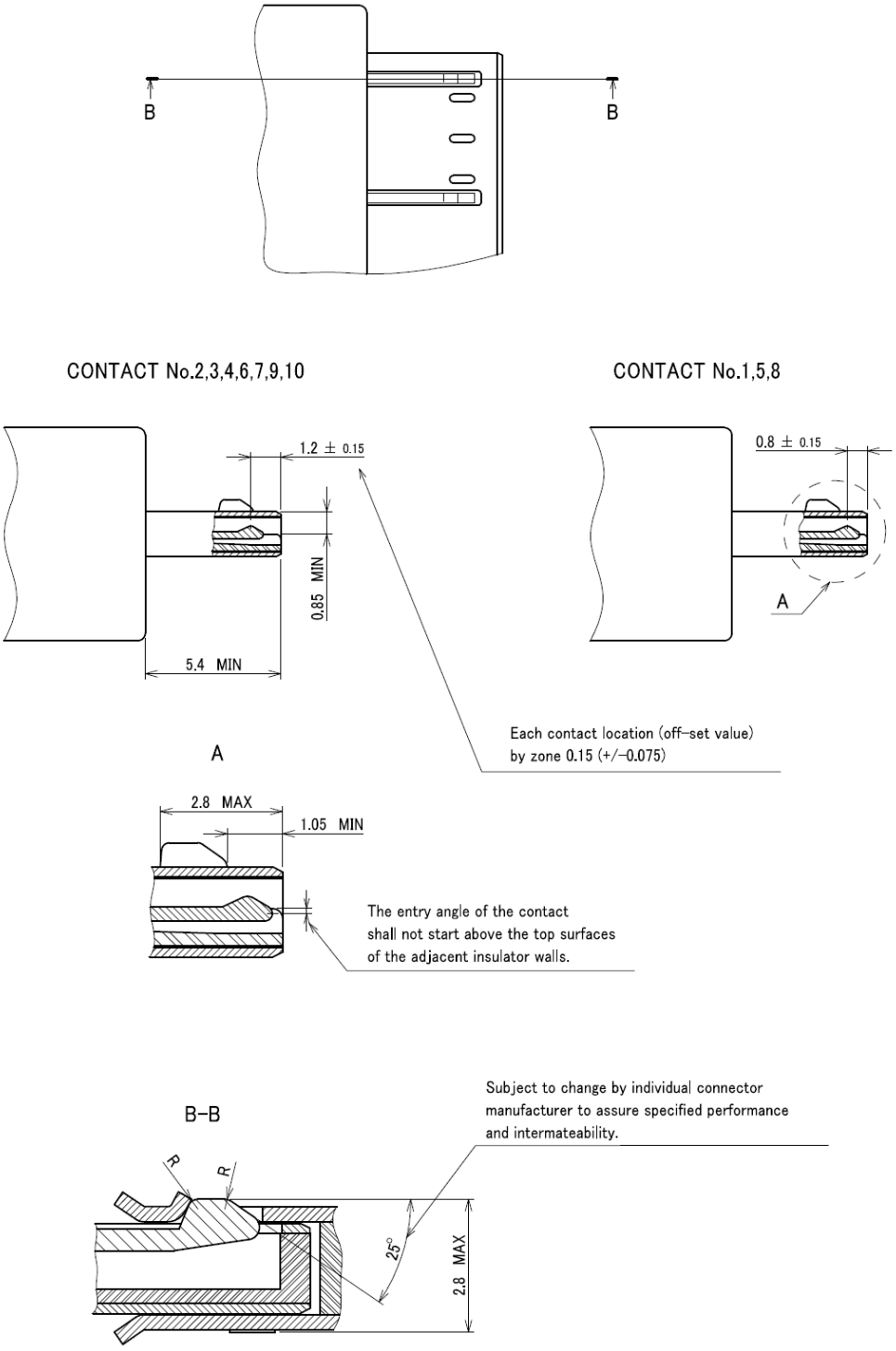
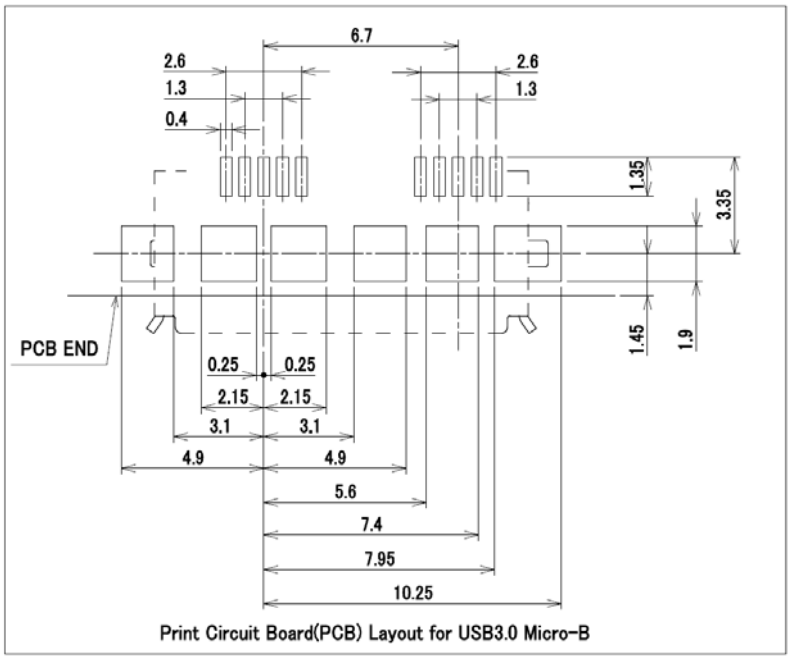
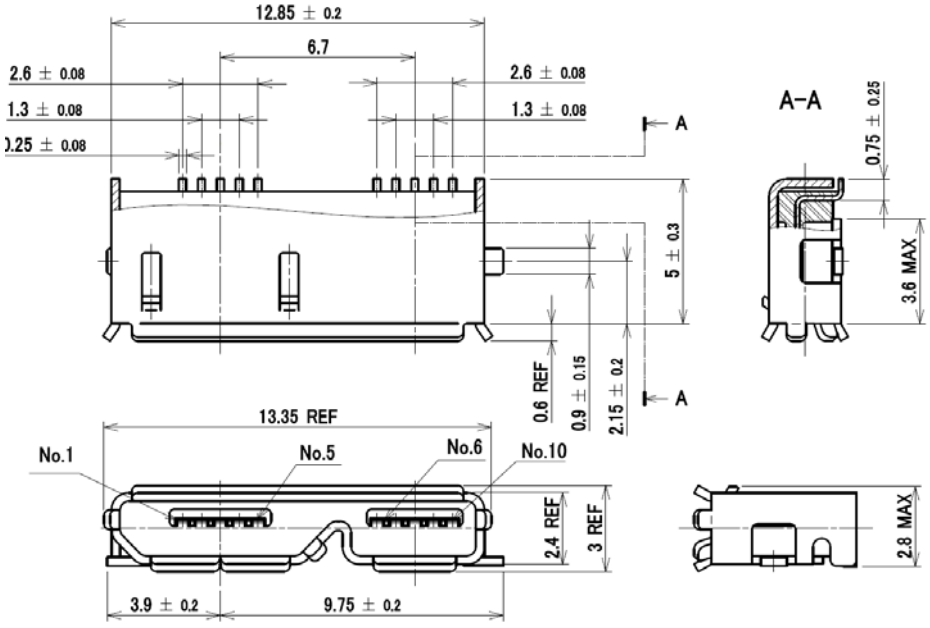


Figure 5-13. USB 3.0 Micro-B and Micro-A Plug Interface Dimensions



Print Circuit Board(PCB) Layout for USB3.0 Micro-B

Standard-Surface Mount-Version Drawing

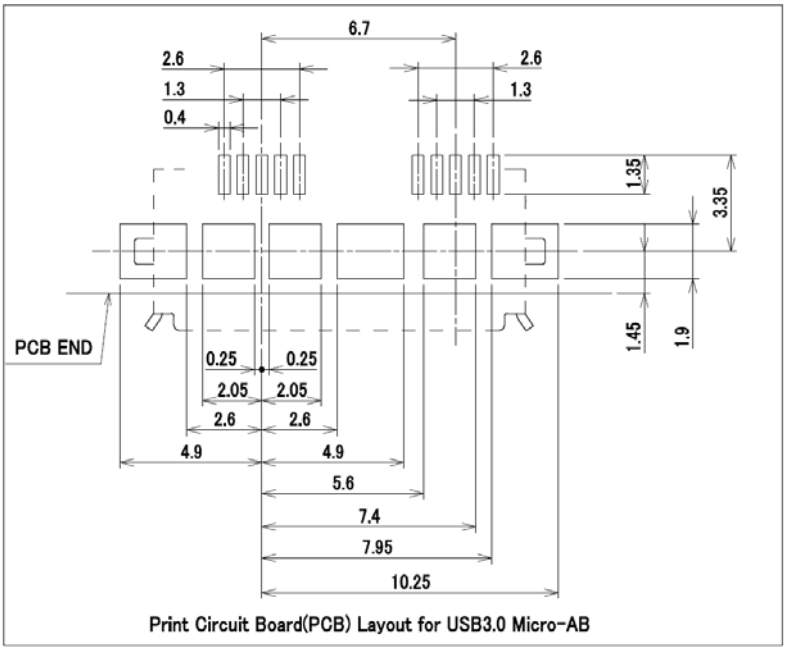


USB3.0 Micro-B RECEPTACLE

NOTES :

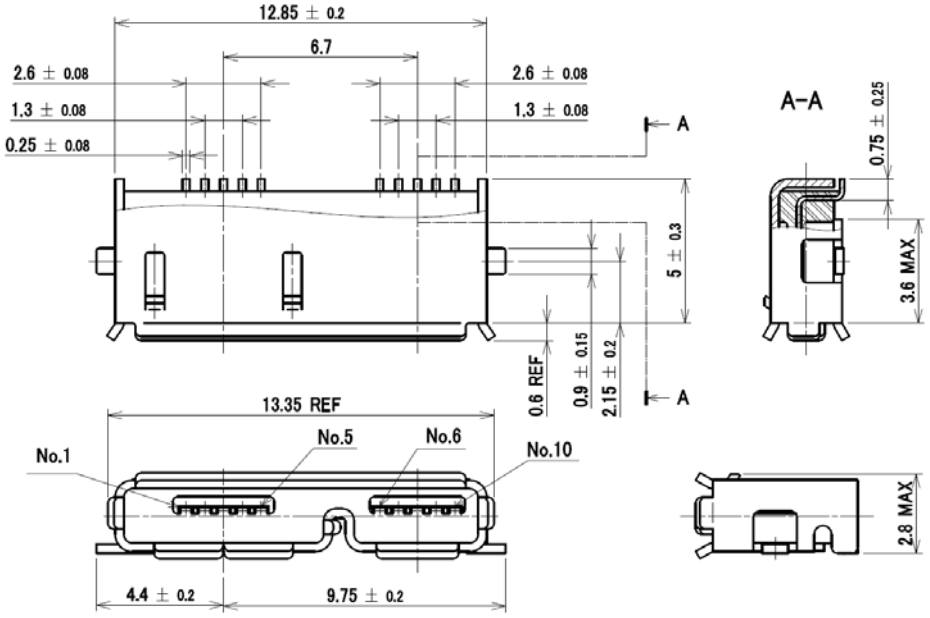
1. Critical Dimensions are TOLERANCED and shall not be deviated.
2. Dimensions that labeled REF are typical dimensions and may vary from manufacturer to manufacturer.
3. General tolerance is +/- 0.05 mm, otherwise the specified tolerances apply.
4. Chamfer metals are optional with no sharp edges.

continued



Print Circuit Board(PCB) Layout for USB3.0 Micro-AB

Standard-Surface Mount-Version Drawing



USB3.0 Micro-AB RECEPTACLE

NOTES :

1. Critical Dimensions are TOLERANCED and shall not be deviated.
2. Dimensions that labeled REF are typical dimensions and may vary from manufacturer to manufacturer.
3. General tolerance is +/- 0.05 mm, otherwise the specified tolerances apply.
4. Chamfer metals are optional with no sharp edges.

Figure 5-14. Reference Footprint for the USB 3.0 Micro-B or Micro-AB Receptacle

The USB 3.0 Micro connector family has the following characteristics:

- The USB 3.0 Micro-B connector may be considered a combination of USB 2.0 Micro-B interface and the USB 3.0 SuperSpeed contacts. The USB 3.0 Micro-B receptacle accepts a USB 2.0 Micro-B plug, maintaining backward compatibility.
- The USB 3.0 Micro-B connector maintains the same connector height and contact pitch as the USB 2.0 Micro-B connector.
- The USB 3.0 Micro-B connector uses the same, proven latch design as the USB 2.0 Micro-B connector.
- The USB 3.0 Micro-AB receptacle is identical to the USB 3.0 Micro-B receptacle except for a keying difference in the connector shell outline.
- The USB 3.0 Micro-A plug is similar to the USB 3.0 Micro-B plug with different keying and ID pin connections. Section 5.3.4.2 discusses the ID pin connections.
- There is no required footprint for the USB 3.0 Micro connector family. Figure 5-14 shows reference Micro-B and -AB connector footprints.

5.3.4.2 Pin Assignments and Description

Table 5-5 and Table 5-6 show the pin assignments for the USB 3.0 Micro connector family.

Table 5-5. USB 3.0 Micro-B Connector Pin Assignments

Pin Number	Signal Name	Description	Mating Sequence
1	VBUS	Power	Second
2	D-	USB 2.0 differential pair	Last
3	D+		
4	ID	OTG identification	
5	GND	Ground for power return	Second
6	MicB_SSTX-	SuperSpeed transmitter differential pair	Last
7	MicB_SSTX+		
8	GND_DRAIN	Ground for SuperSpeed signal return	Second
9	MicB_SSRX-	SuperSpeed receiver differential pair	Last
10	MicB_SSRX+		
Shell	Shield	Connector metal shell	First

Note: Tx and Rx are defined from the device perspective

Table 5-6. USB 3.0 Micro-AB/-A Connector Pin Assignments

Pin Number	Signal Name	Description	Mating Sequence
1	VBUS	Power	Second
2	D-	USB 2.0 differential pair	Last
3	D+		
4	ID	OTG identification	
5	GND	Ground for power return	Second
6	MicA_SSRX-	SuperSpeed receiver differential pair	Last
7	MicA_SSRX+		
8	GND_DRAIN	Ground for SuperSpeed signal return	Second
9	MicA_SSTX-	SuperSpeed transmitter differential pair	Last
10	MicA_SSTX+		
Shell	Shield	Connector metal shell	First

Note: Tx and Rx are defined when an OTG device serves as a host.

The physical location of the pins in the connector is illustrated in Figure 5-12 and Figure 5-14.

5.4 Cable Construction and Wire Assignments

This section discusses the USB 3.0 cables, including cable construction, wire assignments, and wire gauges. The performance requirements will be specified in Section 5.6.1.1.

5.4.1 Cable Construction

Figure 5-15 illustrates a USB 3.0 cable cross-section. There are three groups of wires: UTP signal pair, Shielded Differential Pair (SDP, twisted or twinax signal pairs), and power and ground wires.

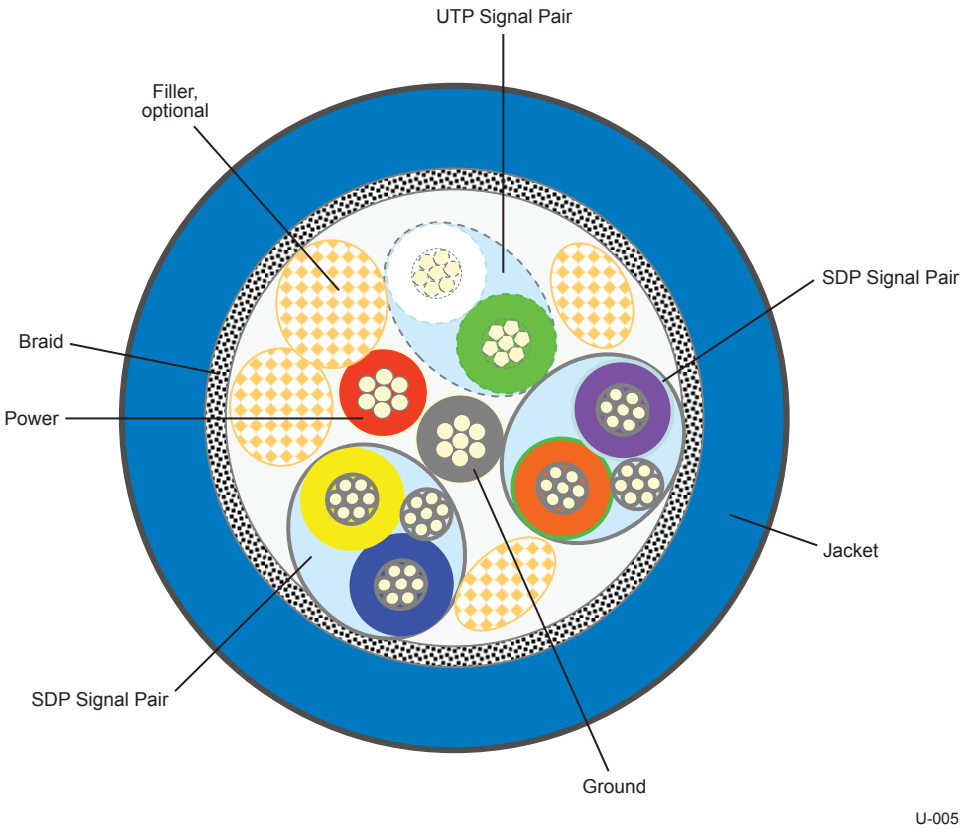


Figure 5-15. Illustration of a USB 3.0 Cable Cross-Section

The UTP is intended to transmit the USB 2.0 signaling while the SDPs are used for SuperSpeed; the shield is needed for the SuperSpeed differential pairs for signal integrity and EMI performance. Each SDP is attached with a drain wire, which is eventually connected to the system ground through the GND_DRAIN pin(s) in the connector.

A metal braid is required to enclose all the wires in the USB 3.0 cable. The braid is to be terminated to the plug metal shells, as close to 360° as possible, to contain EMI.

5.4.2 Wire Assignments

Table 5-7 defines the wire number, signal assignments, and colors of the wires.

Table 5-7. Cable Wire Assignments

Wire Number	Signal Name	Description	Color
1	PWR	Power	Red
2	UTP_D-	Unshielded twist pair, negative	White
3	UTP_D+	Unshielded twist pair, positive	Green
4	GND_PWRrt	Ground for power return	Black
5	SDP1-	Shielded differential pair 1, negative	Blue
6	SDP1+	Shielded differential pair 1, positive	Yellow
7	SDP1_Drain	Drain wire for SDP1	
8	SDP2-	Shielded differential pair 2, negative	Purple
9	SDP2+	Shielded differential pair 2, positive	Orange
10	SDP2_Drain	Drain wire for SDP2	
Braid	Shield	Cable external braid to be 360° terminated on to plug metal shell	

5.4.3 Wire Gauges and Cable Diameters

This specification chooses not to specify wire gauges. Table 5-8 lists the typical wire gauges *for reference*. A large gauge wire incurs less loss, but at the cost of cable flexibility. One should choose the smallest possible wire gauges that meet the cable assembly electrical requirements.

To maximize cable flexibility, all wires are required to be stranded and the cable outer diameter should be minimized as much as possible. A typical USB 3.0 cable outer diameter may range from 3 mm to 6 mm.

Table 5-8. Reference Wire Gauges

Wire Number	Signal Name	Wire Gauge (AWG)
1	PWR	20-28
2	UTP_D-	28-34
3	UTP_D+	28-34
4	GND_PWRrt	20-28
5	SDP1-	26-34
6	SDP1+	26-34
7	SDP1_Drain	28-34
8	SDP2-	26-34
9	SDP2+	26-34
10	SDP2_Drain	28-34

5.5 Cable Assemblies

5.5.1 USB 3.0 Standard-A to USB 3.0 Standard-B Cable Assembly

Figure 5-16 shows a USB 3.0 Standard-A to USB 3.0 Standard-B cable assembly.

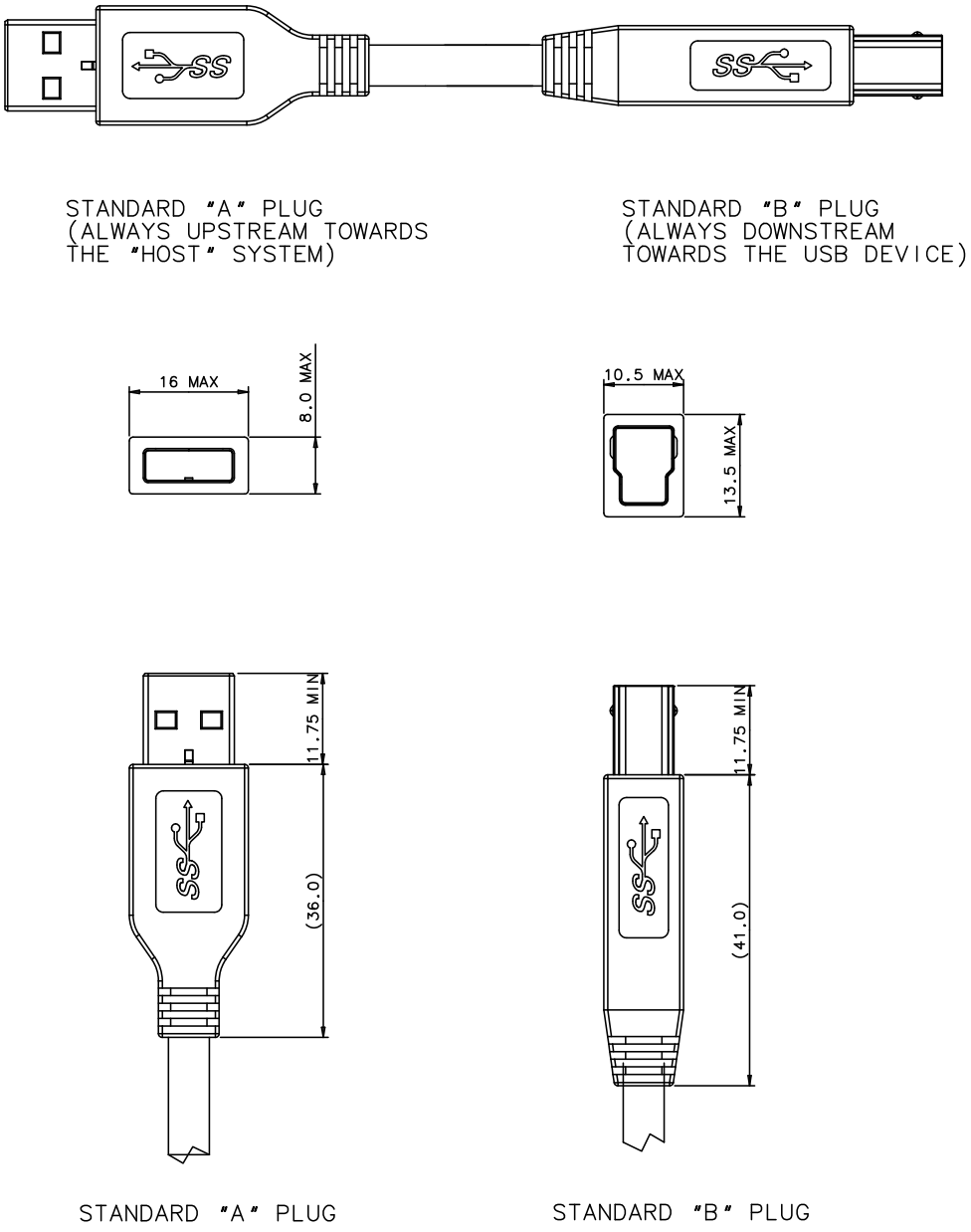


Figure 5-16. USB 3.0 Standard-A to USB 3.0 Standard-B Cable Assembly

Table 5-9 defines the wire connections for the USB 3.0 Standard-A to USB 3.0 Standard-B cable assembly.

Table 5-9. USB 3.0 Standard-A to USB 3.0 Standard-B Cable Assembly Wiring

USB 3.0 Standard-A Plug		Wire		USB 3.0 Standard-B Plug	
Pin Number	**Signal Name**	**Wire Number**	**Signal Name**	**Pin Number**	**Signal Name**
1	VBUS	1	PWR	1	VBUS
2	D-	2	UTP_D-	2	D-
3	D+	3	UTP_D+	3	D+
4	GND	4	GND_PWRrt	4	GND
5	StdA_SSRX-	5	SDP1-	5	StdB_SSTX-
6	StdA_SSRX+	6	SDP1+	6	StdB_SSTX+
7	GND_DRAIN	7 and 10	SDP1_Drain SDP2_Drain	7	GND_DRAIN
8	StdA_SSTX-	8	SDP2-	8	StdB_SSRX-
9	StdA_SSTX+	9	SDP2+	9	StdB_SSRX+
Shell	Shield	Braid	Shield	Shell	Shield

5.5.2 USB 3.0 Standard-A to USB 3.0 Standard-A Cable Assembly

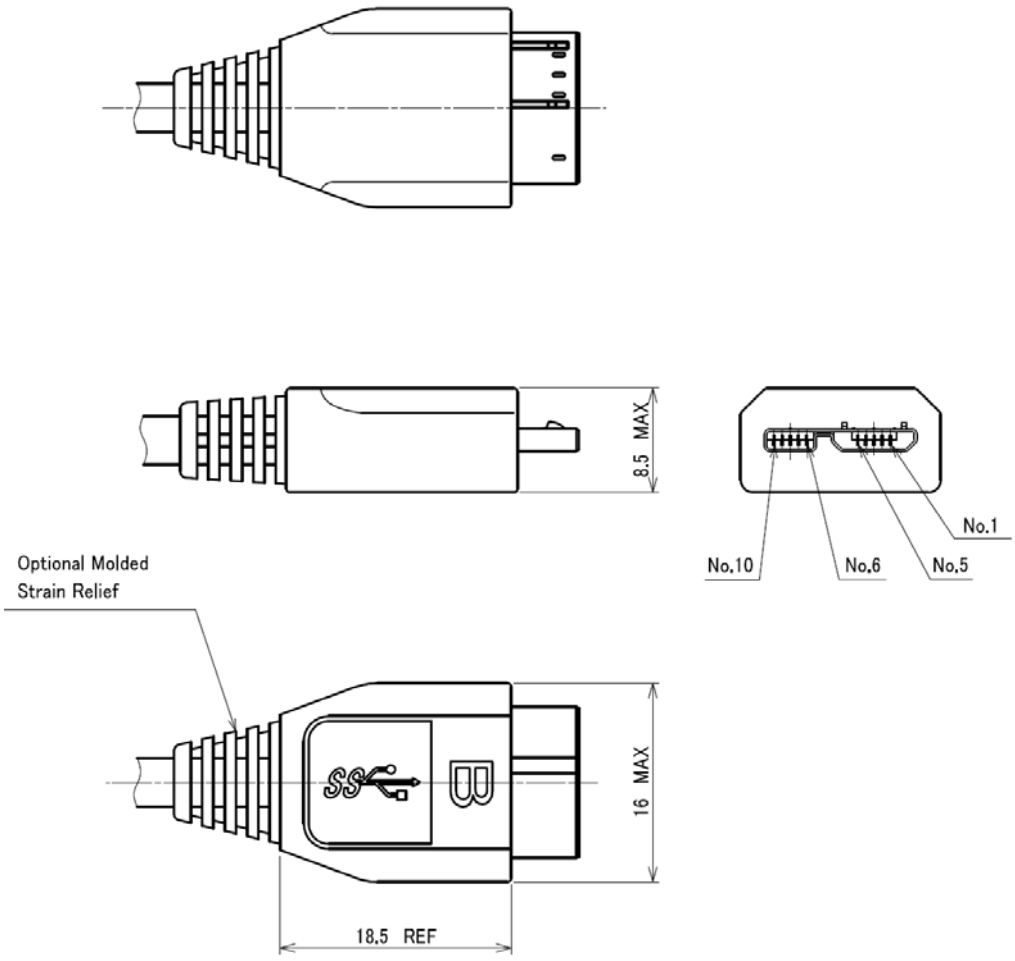
The USB 3.0 Standard-A to USB 3.0 Standard-A cable assembly is defined for operating system debugging and other host-to-host connection applications. Table 5-10 shows wire connections for such a cable assembly. Refer to Figure 5-16 for the USB 3.0 Standard-A plug cable overmold dimensions.

Table 5-10. USB 3.0 Standard-A to USB 3.0 Standard-A Cable Assembly Wiring

USB 3.0 Standard-A Plug #1		Wire		USB 3.0 Standard-A Plug #2	
Pin Number	**Signal Name**	**Wire Number**	**Signal Name**	**Pin Number**	**Signal Name**
1	VBUS	No connect		1	VBUS
2	D-	No connect		2	D-
3	D+	No connect		3	D+
4	GND	4	GND_PWRrt	4	GND
5	StdA_SSRX-	5	SDP1-	8	StdA_SSTX-
6	StdA_SSRX+	6	SDP1+	9	StdA_SSTX+
7	GND_DRAIN	7 & 10	SDP1_Drain SDP2_Drain	7	GND_DRAIN
8	StdA_SSTX-	8	SDP2-	5	StdA_SSRX-
9	StdA_SSTX+	9	SDP2+	6	StdA_SSRX+
Shell	Shield	Braid	Shield	Shell	Shield

5.5.3 USB 3.0 Standard-A to USB 3.0 Micro-B Cable Assembly

Figure 5-17 shows the USB 3.0 Micro-B plug overmold dimensions for a USB 3.0 Standard-A to USB 3.0 Micro-B cable assembly. The USB 3.0 Standard-A plug overmold dimensions can be found in Figure 5-16.



NOTES:
1. Any surface can have texturing up to 0.3mm below the surface.
2. A square area around the letter 'B' can be lowered by as much as 0.5mm.
3. USB authorized logo mark, connector type letter designation– (A or B), color of the insulator body and maximum dimensions are mandatory. Overmolding outer configuration, color and final shape are reference.
4. Pin 4 is not connected to pin 5 inside the plug.

Figure 5-17. USB 3.0 Micro-B Plug Cable Overmold Dimensions

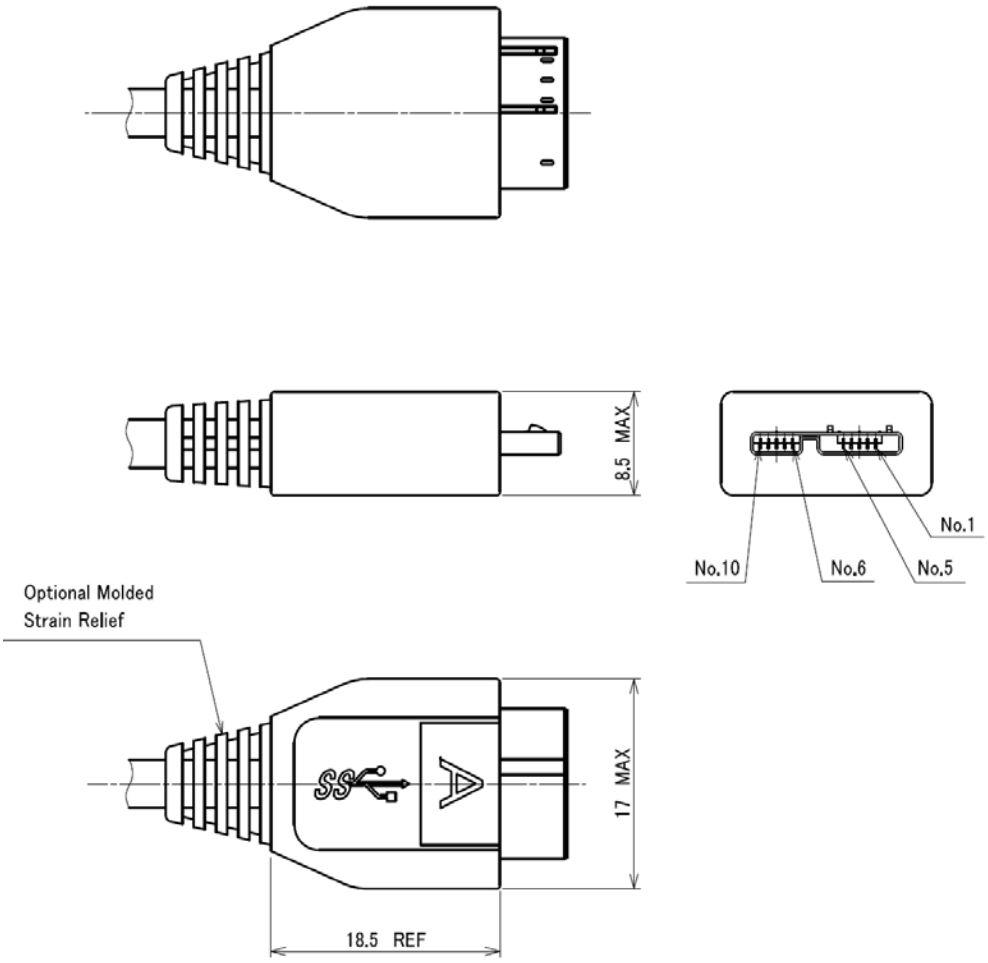
Table 5-11 shows the wire connections for the USB 3.0 Standard-A to USB 3.0 Micro-B cable assembly. Note that the ID pin in the USB 3.0 Micro-B plug shall not be connected, but left in the open condition.

Table 5-11. USB 3.0 Standard-A to USB 3.0 Micro-B Cable Assembly Wiring

USB 3.0 Standard-A Plug		Wire		USB 3.0 Micro-B Plug	
Pin Number	**Signal Name**	**Wire Number**	**Signal Name**	**Pin Number**	**Signal Name**
1	VBUS	1	PWR	1	VBUS
2	D-	2	UTP_D-	2	D-
3	D+	3	UTP_D+	3	D+
4	GND	4	GND_PWRrt	5	GND
5	StdA_SSRX-	5	SDP1-	6	MicB_SSTX-
6	StdA_SSRX+	6	SDP1+	7	MicB_SSTX+
7	GND_DRAIN	7 and 10	SDP1_Drain SDP2_Drain	8	GND_DRAIN
8	StdA_SSTX-	8	SDP2-	9	MicB_SSRX-
9	StdA_SSTX+	9	SDP2+	10	MicB_SSRX+
				4	ID
Shell	Shield	Braid	shield	Shell	Shield

5.5.4 USB 3.0 Micro-A to USB 3.0 Micro-B Cable Assembly

Figure 5-18 shows the USB 3.0 Micro-A plug cable overmold dimensions in a USB 3.0 Micro-A to USB 3.0 Micro-B cable assembly. The USB 3.0 Micro-B plug cable overmold dimensions are shown in Figure 5-17.



NOTES:

1. Any surface can have texturing up to 0.3mm above the surface.
2. A square area around the letter 'A' can be raised as much as 0.5mm above the surface.
3. USB authorized logo mark, connector type letter designation– (A or B), color of the insulator body and maximum dimensions are mandatory. Overmolding outer configuration, color and final shape are reference.
4. Pin 4 is connected to pin 5 inside the plug.

Figure 5-18. USB 3.0 Micro-A Cable Overmold Dimensions

Table 5-12 shows the wire connections for the USB 3.0 Micro-A to USB 3.0 Micro-B cable assembly. The ID pin on a USB 3.0 Micro-A plug shall be connected to the GND pin. The ID pin on a USB 3.0 Micro-B plug shall be a no-connect or connected to ground by a resistance of greater than Rb_PLUG_ID (1 MΩ minimum). An OTG device is required to be able to detect whether a USB 3.0 Micro-A or USB 3.0 Micro-B plug is inserted by determining if the ID pin resistance to ground is less than Ra_PLUG_ID (10 Ω maximum) or if the resistance to ground is greater than Rb_PLUG_ID. Any ID resistance less than Ra_PLUG_ID shall be treated as ID = FALSE and any resistance greater than Rb_PLUG_ID shall be treated as ID = TRUE.

Table 5-12. USB 3.0 Micro-A to USB 3.0 Micro-B Cable Assembly Wiring

USB 3.0 Micro-A Plug		Wire		USB 3.0 Micro-B Plug	
Pin Number	**Signal Name**	**Wire Number**	**Signal Name**	**Pin Number**	**Signal Name**
1	VBUS	1	PWR	1	VBUS
2	D-	2	UTP_D-	2	D-
3	D+	3	UTP_D+	3	D+
4	ID (see Note 1)	No Connect		4	ID (see Note 2)
5	GND	4	GND_PWRrt	5	GND
6	MicA_SSTX-	5	SDP1-	9	MicB_SSRX-
7	MicA_SSTX+	6	SDP1+	10	MicB_SSRX+
8	GND_DRAIN	7 and 10	SDP1_Drain SDP2_Drain	8	GND_DRAIN
9	MicA_SSRX-	8	SDP2-	6	MicB_SSTX-
10	MicA_SSRX+	9	SDP2+	7	MicB_SSTX+
Shell	Shield	Braid	Shield	Shell	Shield

Notes:

1. Connect to the GND.
2. No connect or connect to ground by a resistance greater than 1 MΩ minimum.

5.5.5 USB 3.0 Micro-A to USB 3.0 Standard-B Cable Assembly

A USB 3.0 Micro-A to USB 3.0 Standard-B cable assembly is also allowed. Figure 5-18 and Figure 5-16 show, respectively, the UBS 3.0 Micro-A cable overmold and the USB 3.0 Standard-B cable overmold dimensions.

Table 5-13 shows the wire connections for the USB 3.0 Micro-A to USB 3.0 Standard-B cable assembly.

Table 5-13. USB 3.0 Micro-A to USB 3.0 Standard-B Cable Assembly Wiring

USB 3.0 Micro-A Plug		Wire		USB 3.0 Standard-B Plug	
Pin Number	**Signal Name**	**Wire Number**	**Signal Name**	**Pin Number**	**Signal Name**
1	VBUS	1	PWR	1	VBUS
2	D-	2	UTP_D-	2	D-
3	D+	3	UTP_D+	3	D+
4	ID (see Note 1)	No Connect			
5	GND	4	GND_PWRrt	4	GND
6	MicA_SSTX-	5	SDP1-	8	StdB_SSRX-
7	MicA_SSTX+	6	SDP1+	9	StdB_SSRX+
8	GND_DRAIN	7 and 10	SDP1_Drain SDP2_Drain	7	GND_DRAIN
9	MicA_SSRX-	8	SDP2-	5	StdB_SSTX-
10	MicA_SSRX+	9	SDP2+	6	StdB_SSTX+
Shell	Shield	Braid	Shield	Shell	Shield

Notes:

1. Connect to the GND

5.5.6 USB 3.0 Icon Location

The USB 3.0 cable assemblies, compliant with the *USB 3.0 Connectors and Cable Assemblies Compliance Specification*, shall display the USB 3.0 Icons illustrated in Figure 5-19.

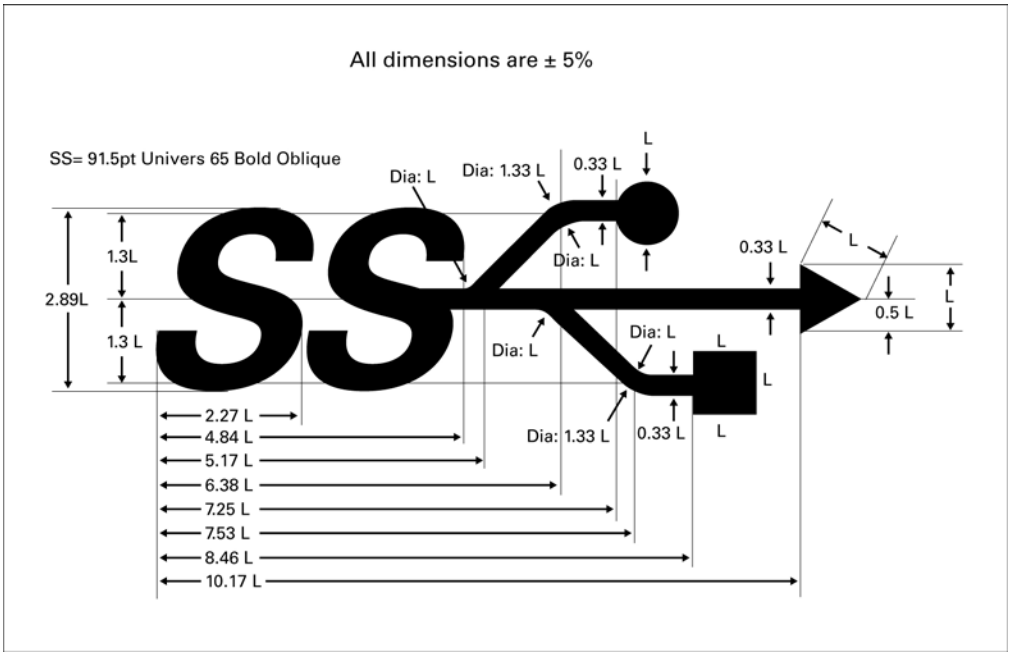


Figure 5-19. USB 3.0 Icon

The USB 3.0 Icon is embossed, in a recessed area, on the side of the USB 3.0 plug. This provides easy user recognition and facilitates alignment during the mating process. The USB Icon and Manufacturer's logo should not project beyond the overmold surface. The USB 3.0 compliant cable assembly is required to have the USB 3.0 Icons on the plugs at both ends, while the manufacturer's logo is recommended. USB 3.0 receptacles should be orientated to allow the Icon on the plug to be visible during the mating process. Figure 5-20 shows a typical plug orientation.

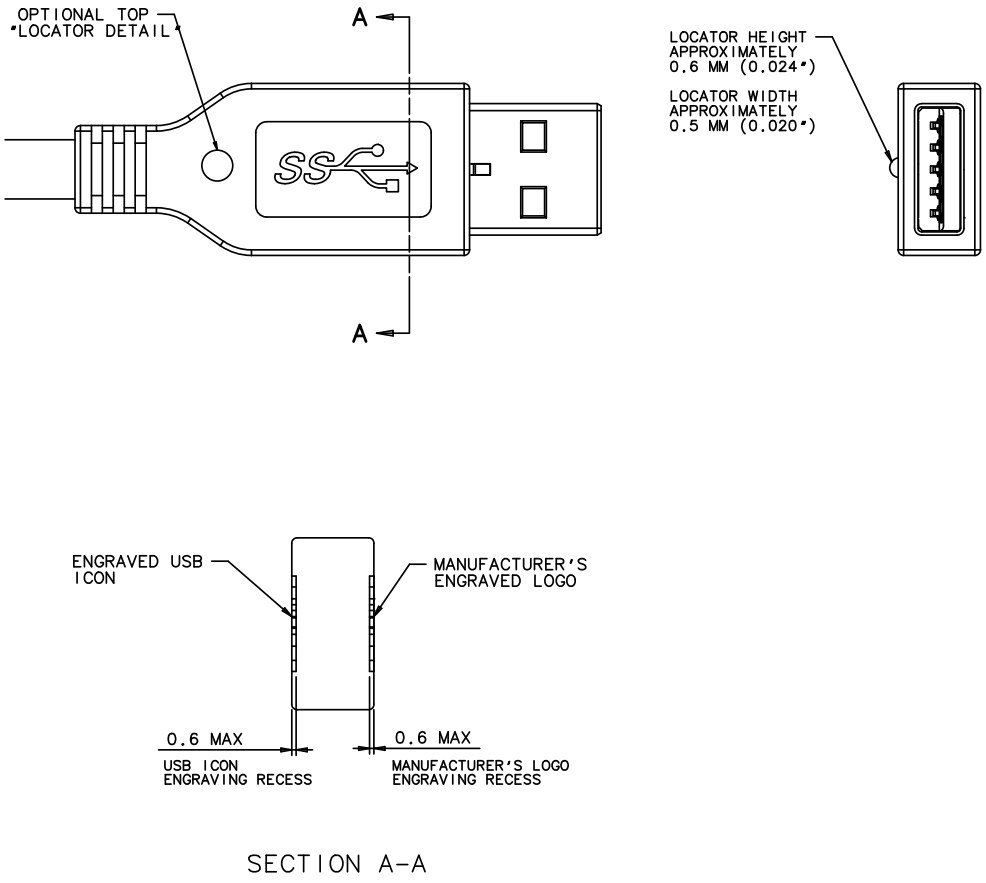


Figure 5-20. Typical Plug Orientation

5.5.7 Cable Assembly Length

This specification does not specify cable assembly lengths. A USB 3.0 cable assembly can be of any length, as long as it meets all the requirements defined in this specification. The cable assembly loss budget defined in Section 5.6.1.3.1 and the cable voltage drop budget defined in Section 11.4.2 will limit the cable assembly length.

5.6 Electrical Requirements

This section covers the electrical requirements for USB 3.0 raw cables, mated connectors, and mated cable assemblies. USB 3.0 signals, known as SuperSpeed, are governed by this specification. The USB 2.0 signals are governed by the USB 2.0 specification, unless otherwise specified. Refer to the *USB 3.0 Connectors and Cable Assemblies Compliance Document* for specific D+/D- lines electrical requirements.

Compliance to the USB 3.0 specification is established through normative requirements of mated connectors and mated cable assemblies. SuperSpeed requirements are specified mainly in terms of S-parameters, using industry test specification with supporting details when required. DC requirements, such as contact resistance and current carrying capability, are also specified in this section.

Any informative specification for cable and connector products is for the purpose of design guidelines and manufacturing control.

In conjunction with performance requirements, the required test method is referenced for the parameter stated. A list of the industry standards for DC requirements is found in the Section 5.6.2. Additional supporting test procedures can be found in the *USB 3.0 Connectors and Cable Assemblies Compliance Document.*

The requirements in the section apply to all USB 3.0 connectors and/or cable assemblies unless specified otherwise.

5.6.1 SuperSpeed Electrical Requirements

The following sections outline the requirements for SuperSpeed signals. The requirements for the USB 2.0 signals (D+/D- lines) are given in the *USB 3.0 Connectors and Cable Assemblies Compliance Document*.

5.6.1.1 Raw Cable

Informative raw cable electrical performance targets are provided here to help cable assembly manufacturers manage raw cable suppliers. *Those targets are not part of the USB 3.0 compliance items; the ultimate requirements will be the mated cable assembly performance specified in Section 5.6.1.3.*

5.6.1.1.1 Characteristic Impedance

The differential characteristic impedance for the SDP pairs is recommended to be within 90 Ω +/- 7 Ω. It should be measured with a TDR in a differential mode using a 200 ps (10%-90%) rise time.

5.6.1.1.2 Intra-Pair Skew

The intra-pair skew for the SDP pairs is recommended to be less than 15 ps/m. It should be measured with a TDT in a differential mode using a 200 ps (10%-90%) rise time with a crossing at 50% of the input voltage.

5.6.1.1.3 Differential Insertion Loss

Cable loss depends on wire gauges and dielectric materials. Table 5-14 lists the examples of differential insertion losses for the SDP pairs. Note that the differential loss values are referenced to a 90 Ω differential impedance.

Table 5-14. SDP Differential Insertion Loss Examples

Frequency	34AWG	30AWG	28AWG	26AWG
0.625 GHz	2.7 dB/m	1.3 dB/m	1.0 dB/m	0.9 dB/m
1.25 GHz	3.3 dB/m	1.9 dB/m	1.5 dB/m	1.3 dB/m
2.50 GHz	4.4 dB/m	3.0 dB/m	2.5 dB/m	1.9 dB/m
5.00 GHz	6.7 dB/m	4.6 dB/m	3.6 dB/m	3.1 dB/m
7.50 GHz	9.0 dB/m	5.9 dB/m	4.7 dB/m	4.2 dB/m

5.6.1.2 Mated Connector

The mated connector impedance requirement is needed to maintain signal integrity. The differential impedance of a mated connector shall be within 90 Ω +/-15 Ω, as seen from a 50 ps (20%-80%) risetime of a differential TDR. Figure 5-21 illustrates the impedance limits of a mated connector. The impedance profile of a mated connector must fall within the limits shown in Figure 5-21. Note that the impedance profile of the mated connector is defined from the receptacle footprints through the plug cable termination area. In the case the plug is directly attached to a device PCB, the mated connector impedance profile includes the path from the receptacle footprints to the plug footprints.

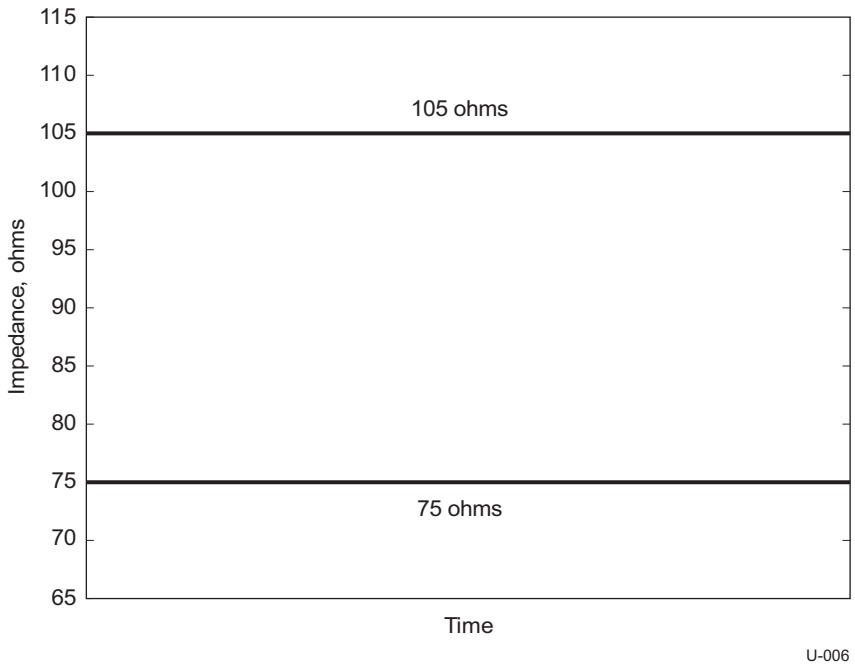


Figure 5-21. Impedance Limits of a Mated Connector

5.6.1.3 Mated Cable Assemblies

A mated cable assembly refers to a cable assembly mated with the corresponding receptacles mounted on a test fixture at the both ends. The requirements are for the entire signal path of the mated cable assembly, from the host receptacle contact solder pads or through-holes on the host system board to the device receptacle contact solder pads or through holes on the device system board, not including PCB traces, as illustrated in Figure 5-22; the measurement is between TP1 (test point 1) and TP2 (test point 2).

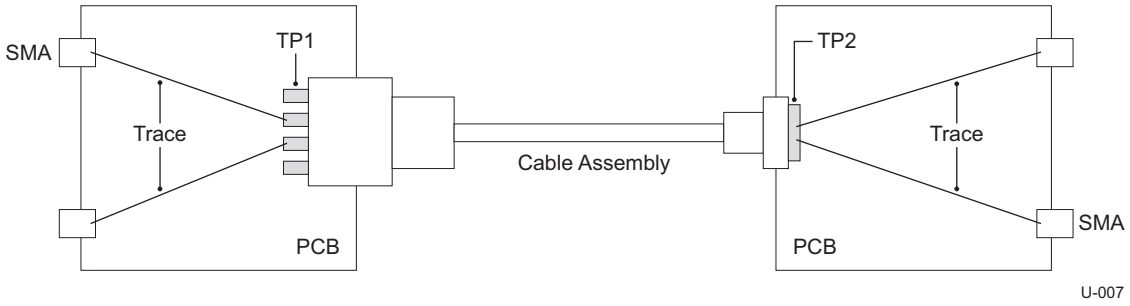


Figure 5-22. Illustration of Test Points for a Mated Cable Assembly

For proper measurements, the receptacles shall be mounted on a test fixture. The test fixture shall have uncoupled access traces from SMA or microprobe launches to the reference planes or test points, preferably with 50 Ω +/-7% Ω single-ended characteristic impedance. The test fixture shall have appropriate calibration structures to calibrate out the fixturing effect. All non-ground pins that are adjacent but not connected to measurement ports shall be terminated with 50 Ω loads.

A reference USB 3.0 mated cable assembly test fixture is defined in the *USB 3.0 Connectors and Cable Assemblies Compliance Document*, in which the detailed testing procedures are given.

5.6.1.3.1 Differential Insertion Loss (EIA-360-101)

The differential insertion loss, SDD12, measures the differential signal energy transmitted through the mated cable assembly. Figure 5-23 shows the differential insertion loss limit, which is defined by the following vertices: (100 MHz, -1.5 dB), (1.25 GHz, -5.0 dB), (2.5 GHz, -7.5 dB), and (7.5 GHz, -25 dB). The measured differential insertion loss of a mated cable assembly must not exceed the differential insertion loss limit.

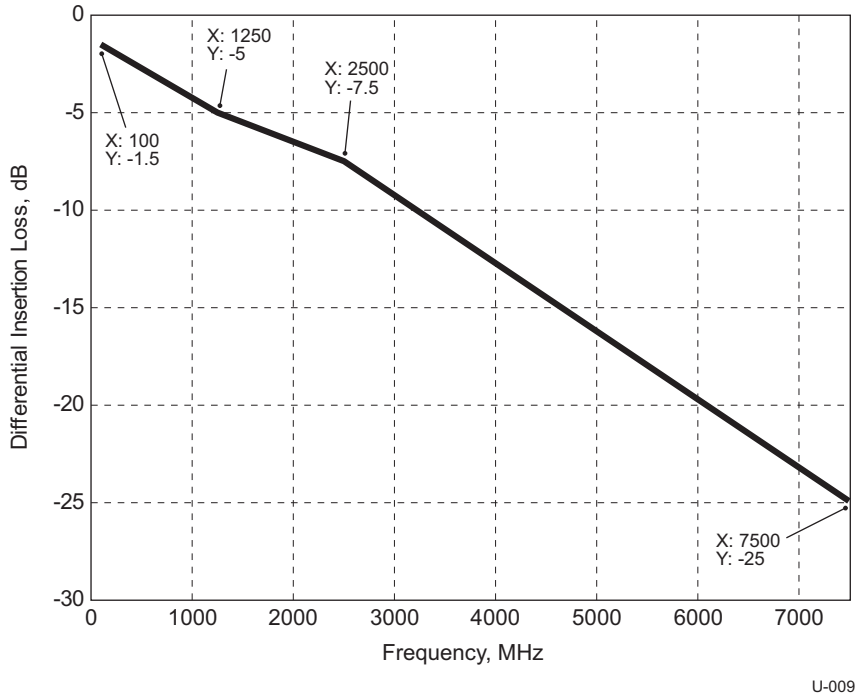


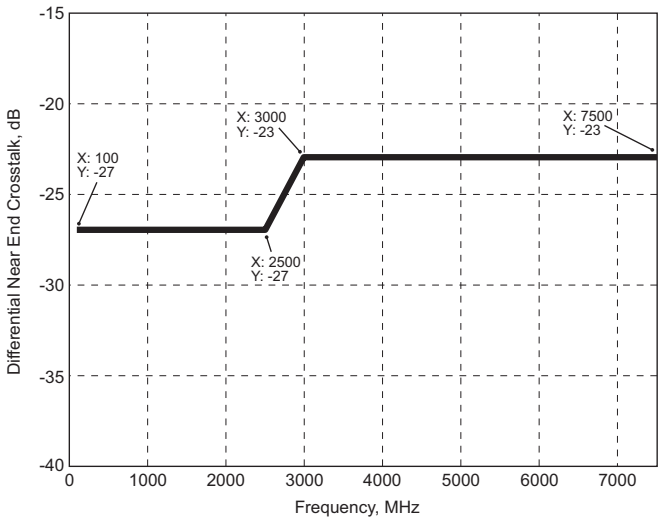
Figure 5-23. Differential Insertion Loss Requirement

5.6.1.3.2 Differential Near-End Crosstalk Between SuperSpeed Pairs (EIA-360-90)

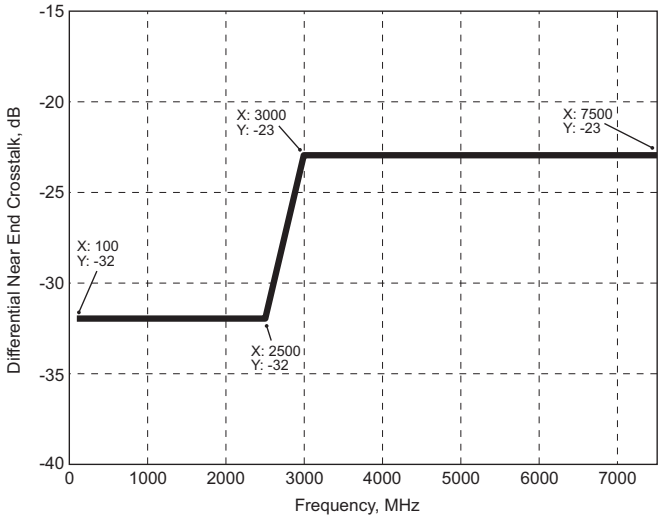
The differential crosstalk measures the unwanted coupling between differential pairs. Since the Tx pair is right next to the Rx pair for SuperSpeed, only the differential near-end crosstalk (DDNEXT) is specified, as shown in Figure 5-24. The mated cable assembly meets the DDNEXT requirement if its DDNEXT does not exceed the limit shown in Figure 5-24; the vertices that defines the DDNEXT limit are:

- For the USB 3.0 Micro connector family: (100 MHz, -27 dB), (2.5 GHz, -27 dB), (3 GHz, -23 dB), and (7.5 GHz, -23 dB)
- For all other USB 3.0 connectors: (100 MHz, -32 dB), (2.5 GHz, -32 dB), (3 GHz,-23 dB), and (7.5 GHz, -23 dB)

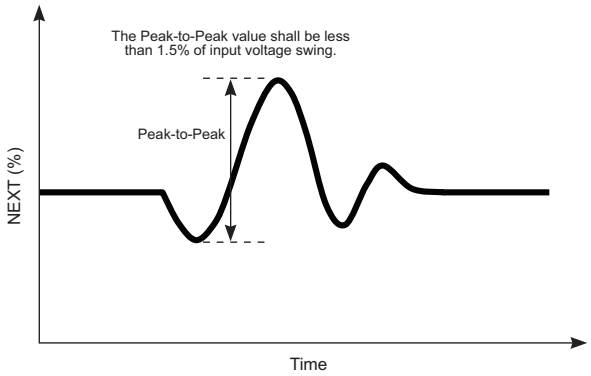
For the USB 3.0 Standard-B and Powered-B connectors: If a Standard-B or Powered-B connector fails to meet the -32 dB requirement in frequency domain shown in Figure 5-24(b), but is able to meet the -27 dB requirement shown in Figure 5-24(a), the measurement of DDNEXT in time domain will be required. The peak-to-peak value of DDNEXT in time domain shall not exceeds 1.5% of the input voltage swing (as shown in Figure 5-24(c)), as seen from a 50 ps (20%-80%) rise time of a differential Time Domain Transmission (TDT) at TP1 or TP2 (as shown in Figure 5-22). Note that removing fixture effect is not required for the time domain measurements. If a Standard-B or Powered-B connector fails to meet the -27 dB requirement shown in Figure 5-24(a), it shall be considered non-compliant.



(a) For USB 3.0 Micro-B Family



(b) For all other USB 3.0 Connectors



(c) For USB 3.0 Standard-B and Powered-B Connectors in Time Domain

U-010A

Figure 5-24. Differential Near-End Crosstalk Requirement Between SuperSpeed Pairs

5.6.1.3.3 Differential Crosstalk Between D+/D- and SuperSpeed Pairs (EIA-360-90)

The differential near-end and far-end crosstalk between the D+/D- pair and the SuperSpeed pairs (SSTX+/SSTX- or SSRX+/SSRX-) shall be managed not to exceed the limit shown in Figure 5-25; the vertices that defines the DDNEXT and DDFEXT limit are: (100 MHz, -21 dB), (2.5 GHz, -21 dB), (3.0 GHz, -15 dB) and (7.5 GHz, -15 dB).

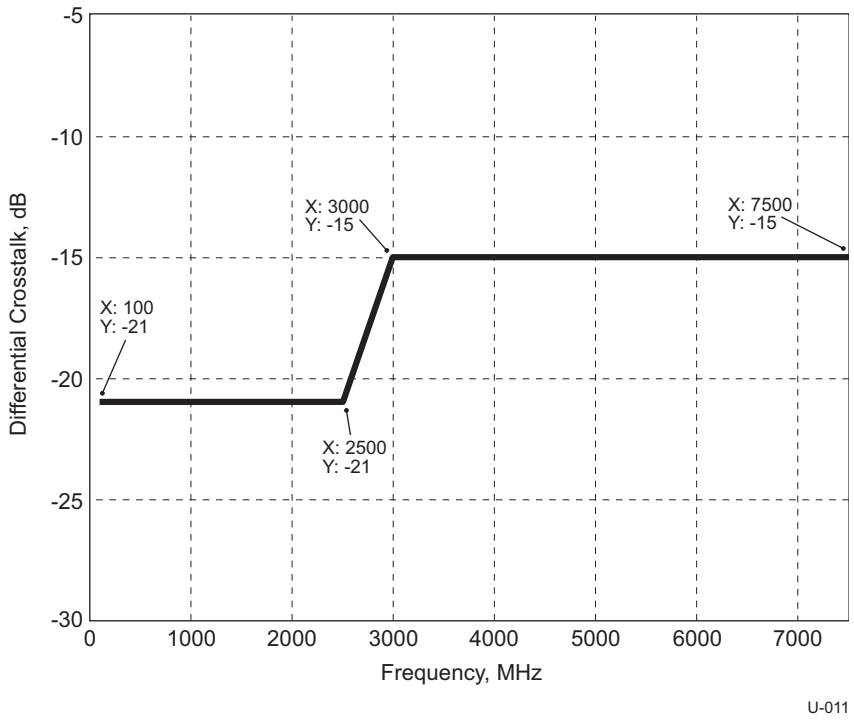


Figure 5-25. Differential Near-End and Far-End Crosstalk Requirement Between D+/D- Pair and SuperSpeed Pairs

5.6.1.3.4 Differential-to-Common-Mode Conversion

Since the common mode current is directly responsible for EMI, limiting the differential-to-common-mode conversion, SCD12, will limit EMI generation within the connector and cable assembly. Figure 5-26 illustrates the SCD12 requirement; a mated cable assembly passes the SCD12 requirement if its SCD12 is less than or equal to -20 dB across the frequency range shown in Figure 5-26.

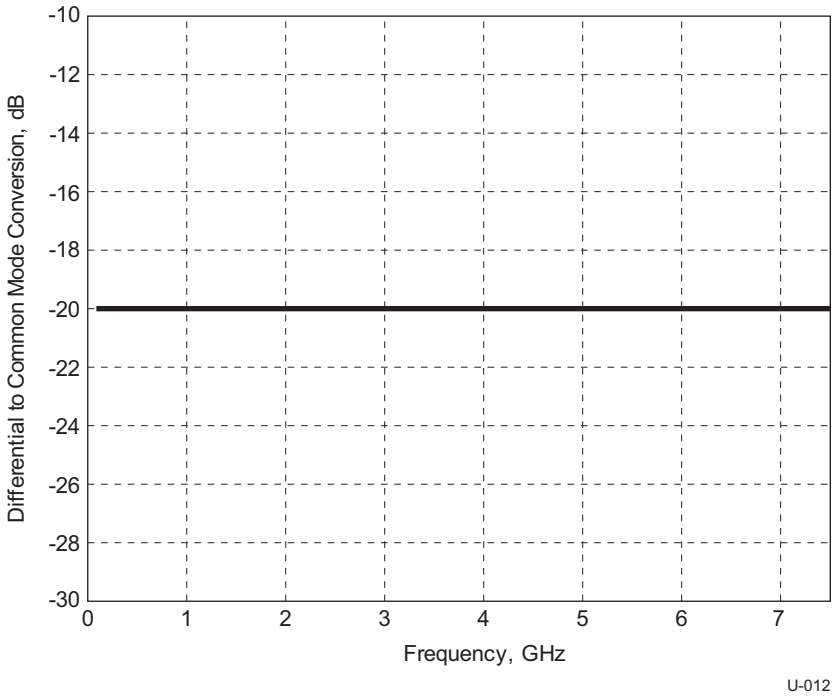


Figure 5-26. Differential-to-Common-Mode Conversion Requirement

5.6.2 DC Electrical Requirements

5.6.2.1 Low Level Contact Resistance (EIA 364-23B)

The following requirement applies to both the power and signal contacts:

- 30 mΩ (Max) initial for VBUS and GND contacts.
- 50 mΩ (Max) initial for all other contacts.
- Maximum change (delta) of +10 mΩ after environmental stresses.
- Measure at 20 mV (Max) open circuit at 100 mA.
- Refer to Section 5.7.2 for environmental requirements and test sequences.

5.6.2.2 Dielectric Strength (EIA 364-20)

No breakdown shall occur when 100 Volts AC (RMS) is applied between adjacent contacts of unmated and mated connectors.

5.6.2.3 Insulation Resistance (EIA 364-21)

A minimum of 100 MΩ insulation resistance is required between adjacent contacts of unmated and mated connectors.

5.6.2.4 Contact Current Rating (EIA 364-70, Method 2)

A current of 1.8 A shall be applied to V_{BUS} pin and its corresponding GND pin (pin 1 and pin 4 of the USB 3.0 Standard-A and Standard-B/Powered-B connectors; pin 1 and pin 5 of the USB 3.0 Micro connector family). Additionally, a minimum current of 0.25 A shall be applied to all the other contacts. When the current is applied to the contacts, the delta temperature shall not exceed +30 °C at any point on the USB 3.0 connectors under test, when measured at an ambient temperature of 25 °C.

In the case of the USB 3.0 Powered-B connector, a current of 2.0 A shall be applied to the DPWR pin and its corresponding DGND pin (pin 10 and pin 11 for USB 3.0 Powered-B connector). Additionally, a minimum current of 0.25 A shall be applied to all the other contacts. When current is applied to the contacts, the delta temperature must not exceed +30 °C at any point in the USB 3.0 connectors under test, when measured at an ambient temperature of 25 °C.

5.7 Mechanical and Environmental Requirements

The requirements in the section apply to all USB 3.0 connectors and/or cable assemblies unless specified otherwise.

5.7.1 Mechanical Requirements

5.7.1.1 Insertion Force (EIA 364-13)

The connector insertion force shall not exceed 35 N at a maximum rate of 12.5 mm (0.492") per minute.

It is recommended to use a non-silicon based lubricant on the latching mechanism to reduce wear. If used, the lubricant may not affect any other characteristic of the system.

5.7.1.2 Extraction Force Requirements (EIA 364-13)

5.7.1.2.1 Extraction Force (EIA 364-13)

The connector extraction force shall not be less than 10 N initial and 8 N after the specified insertion/extraction or durability cycles (at a maximum rate of 12.5 mm (0.492") per minute).

No burs or sharp edges are allowed on top of locking latches (hook surfaces which will rub against the receptacle shield).

It is recommended to use a non-silicon based lubricant on the latching mechanism to reduce wear. If used, the lubricant may not affect any other characteristic of the system.

5.7.1.2.2 Extraction Force (EIA 364-13, USB 3.0 Micro Connector Family Only)

The connector extraction force shall not be less than 10 N or more than 25 N initial and less than 8 N and more than 25 N after the specified insertion/extraction or durability cycles (at a maximum rate of 12.5 mm (0.492") per minute).

No burs or sharp edges are allowed on top of locking latches (hook surfaces which will rub against the receptacle shield).

It is recommended to use a non-silicon based lubricant on the latching mechanism to reduce wear. If used the lubricant may not affect any other characteristic of the system.

5.7.1.3 Durability or Insertion/Extraction Cycles (EIA 364-09)

The durability ratings listed in Table 5-15 are specified for the USB 3.0 connectors.

Table 5-15. Durability Ratings

Connector	Standard Durability Class	High Durability Class
USB 3.0 Standard-A connector	1500 cycles min	5000 cycles min
USB 3.0 Standard-B connector	1500 cycles min	5000 cycles min
USB 3.0 Powered-B connector	1500 cycles min	5000 cycles min
USB 3.0 Micro connector family	10000 cycles min	

The durability test shall be done at a maximum rate of 200 cycles per hour and no physical damage to any part of the connector and cable assembly shall occur.

5.7.1.4 Cable Flexing (EIA 364-41, Condition I)

No physical damage or discontinuity over 1 ms during flexing shall occur to the cable assembly with Dimension X = 3.7 times the cable diameter and 100 cycles in each of two planes.

5.7.1.5 Cable Pull-Out (EIA 364-38, Condition A)

No physical damage to the cable assembly shall occur when it is subjected to a 40 N axial load for a minimum of 1 minute while clamping one end of the cable plug.

5.7.1.6 Peel Strength (USB 3.0 Micro Connector Family Only)

No visible physical damage shall be noticed to a soldered receptacle when it is pulled up from the PCB in the vertical direction with a minimum force of 150 N.

5.7.1.7 4-Axes Continuity Test (USB 3.0 Micro Connector Family Only)

The USB 3.0 Micro connector family shall be tested for continuity under stress using the test configurations shown below. Plugs shall be supplied in a cable assembly with a representative overmold. A USB 3.0 Micro-B or -AB receptacle shall be mounted on a 2-layer printed circuit board (PCB) between 0.8 and 1.0 mm thickness. The PCB shall be clamped on either side of the receptacle no further than 5 mm away from the solder tails. The PCB shall initially be placed in a horizontal plane, and an 8-N tensile force shall be applied to the cable in a downward direction, perpendicular to the axis of insertion, for a period of at least 10 seconds.

The continuity across each contact shall be measured throughout the application of the tensile force. The PCB shall then be rotated 90 degrees such that the cable is still inserted horizontally and the 8 N tensile force will be applied again in the downward direction and continuity measured as before. This test will be repeated for 180-degree and 270-degree rotations. Passing parts shall not exhibit any discontinuities greater than 1 μs duration in any of the four orientations.

One method for measuring the continuity through the contacts is to short all the wires at the end of the cable pigtail and apply a voltage through a pull-up to each of VBUS, D+, D-, ID, and the SuperSpeed pins, with the GND pins connected to ground.

When testing a USB 3.0 Micro-A plug, all the sense resistors shall stay pulled down for the length of the test. When testing a USB 3.0 Micro-B plug, the ID pin shall stay high and the other pins shall remain low for the duration of the test. Alternate methods are allowed to verify continuity through all pins.

The 4-axes continuity tests shall be done with a USB 3.0 Micro-B/-A plug in a USB 3.0 Micro-B/-AB receptacle and with a USB 2.0 Micro-B/-A plug in a USB 3.0 Micro-B/-AB receptacle, as illustrated in Figure 5-27.

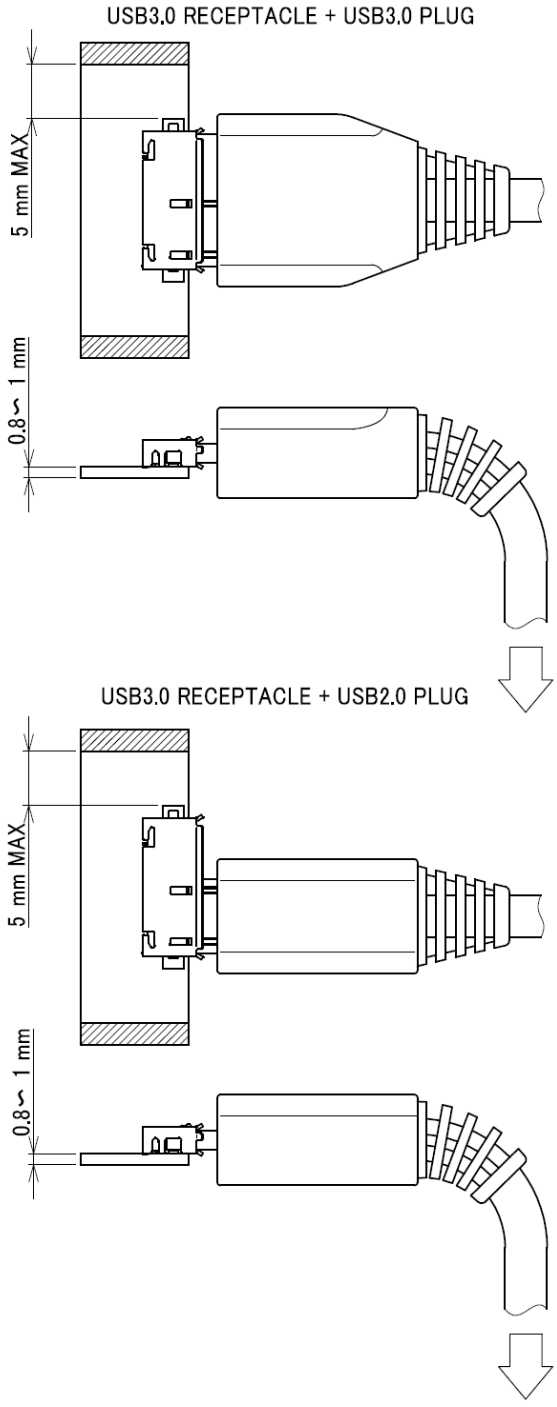


Figure 5-27. 4-Axes Continuity Test

5.7.1.8 Wrenching Strength (Reference, USB 3.0 Micro Connector Family Only)

The wrenching strength test shall be performed using virgin parts. Perpendicular forces (Fp) are applied to a plug when inserted at a distance (L) of 15 mm from the edge of the receptacle. Testing conditions and method shall be agreed to by all parties. These forces shall be applied in all four directions (left, right, up, down). Compliant connectors shall meet the following force thresholds:

- No plug or receptacle damage shall occur when a force of 0-25 N is applied.
- The plug may be damaged, but only in such a way that the receptacle does not sustain damage when a force of 25-50 N is applied.

5.7.1.9 Lead Co-Planarity

Co-planarity of all SMT leads shall be within a 0.08 mm range.

5.7.1.10 Solderability

Solder shall cover a minimum of 95% of the surface being immersed, when soldered at a temperature 255 °C +/-5 °C for an immersion duration of 5 s.

5.7.1.11 Restriction of Hazardous Substances (RoHS) Compliance

It is recommended that components be RoHS compliant. Lead-free plug and receptacle materials should conform to Directive 2002/95/EC of January 27, 2003 on RoHS or other regulatory directives.

5.7.2 Environmental Requirements

The connector interface environmental tests shall follow EIA-364-1000.01, Environmental Test Methodology for Assessing the Performance of Electrical Connectors and Sockets Used in Business Office Applications.

Since the connector defined has far more than 0.127 mm wipe length, Test Group 6 in EIA-364-1000.01 is not required. The temperature life test duration and the mixed flowing gas test duration values are derived from EIA 364-1000.01 based on the field temperature per the following.

Table 5-16. Environmental Test Conditions

Temperature Life test temperature and duration	105 °C for 120 hours
Temperature Life test temperature and duration for preconditioning	105 °C for 72 hours
Mixed flowing gas test duration	7 days

The pass/fail criterion for the low level contact resistance (LLCR) is as defined in Section 5.6.2.1. The durability ratings are defined in Section 5.7.1.3.

5.7.3 Materials

This specification does not specify materials for connectors and cables. Connector and cable manufactures shall select appropriate materials based on performance requirements. Table 5-17 below is provided for reference only.

Note:
Connector and cable manufacturers shall comply with contact plating requirements, per the following options:

Option I

Receptacle

Contact area: (Min) 0.05 μm Au + (Min) 0.75 μm Ni-Pd on top of (Min) 2.0 μm Ni

Contact tail: (Min) 0.05 μm Au on top of (Min) 2.0 μm Ni

Plug

Contact area: (Min) 0.05 μm Au + (Min) 0.75 μm Ni-Pd on top of (Min) 2.0 μm Ni

Option II

Receptacle

Contact area: (Min) 0.75 μm Au on top of (Min) 2.0 μm Ni

Contact tail: (Min) 0.05 μm Au on top of (Min) 2.0 μm Ni

Plug

Contact area: (Min) 0.75 μm Au on top of (Min) 2.0 μm Ni.

Other material parameters, which connector and cable manufacturers shall select based on performance parameters, are listed below in Table 5-17 for reference only.

Table 5-17. Reference Materials[1,2]

Component	Materials	Comments
Cable	Conductor: copper with tin plating	
	SDP Shield: AL foil or AL/mylar foil	
	Braid: Tin plated copper or aluminum	
	Jacket: PVC or halogen free substitute material	
Cable Overmold	Thermoset	
Connector Shell	Copper alloy or stainless steel, depending on durability requirement	
Housing	Thermoplastics capable of withstanding lead-free soldering temperature.	The USB 3.0 Standard-A connector housings are recommended to be Pantone 300C (Blue).

[1] Halogen-free materials should be considered for all plastics.

[2] If an application of the part requires a flammability rating less flammable than HB, a suitable flame retardant compound should be considered for plastics in this part.

5.8 Implementation Notes and Design Guides

This section discusses a few implementation notes and design guides to help users design and use the USB 3.0 connectors and cables.

5.8.1 Mated Connector Dimensions

Figure 5-28, Figure 5-29, and Figure 5-30 show the mated plugs and receptacles for the USB 3.0 Standard-A, USB 3.0 Standard-B, and USB 3.0 Micro connectors, respectively. The distance between the receptacle front surface and the cable overmold should be observed by system designers to avoid interference between the system enclosure and the cable plug overmold.

Provisions shall be made in connectors and chassis to ground the connector metal shells to the metal chassis to contain EMI emission.

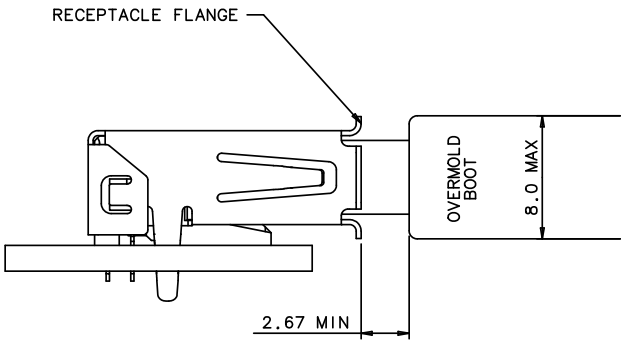


Figure 5-28. Mated USB 3.0 Standard-A Connector

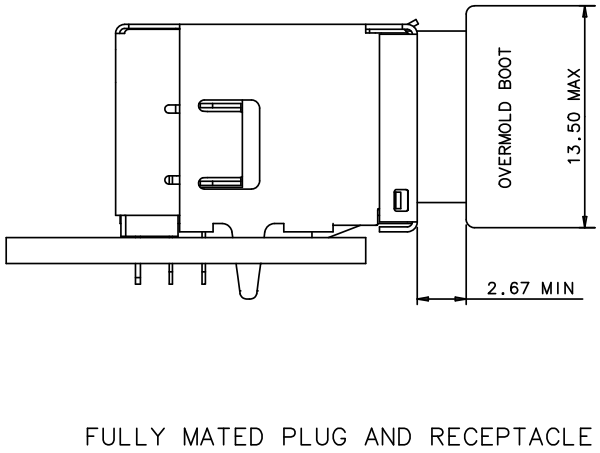


Figure 5-29. Mated USB 3.0 Standard-B Connector

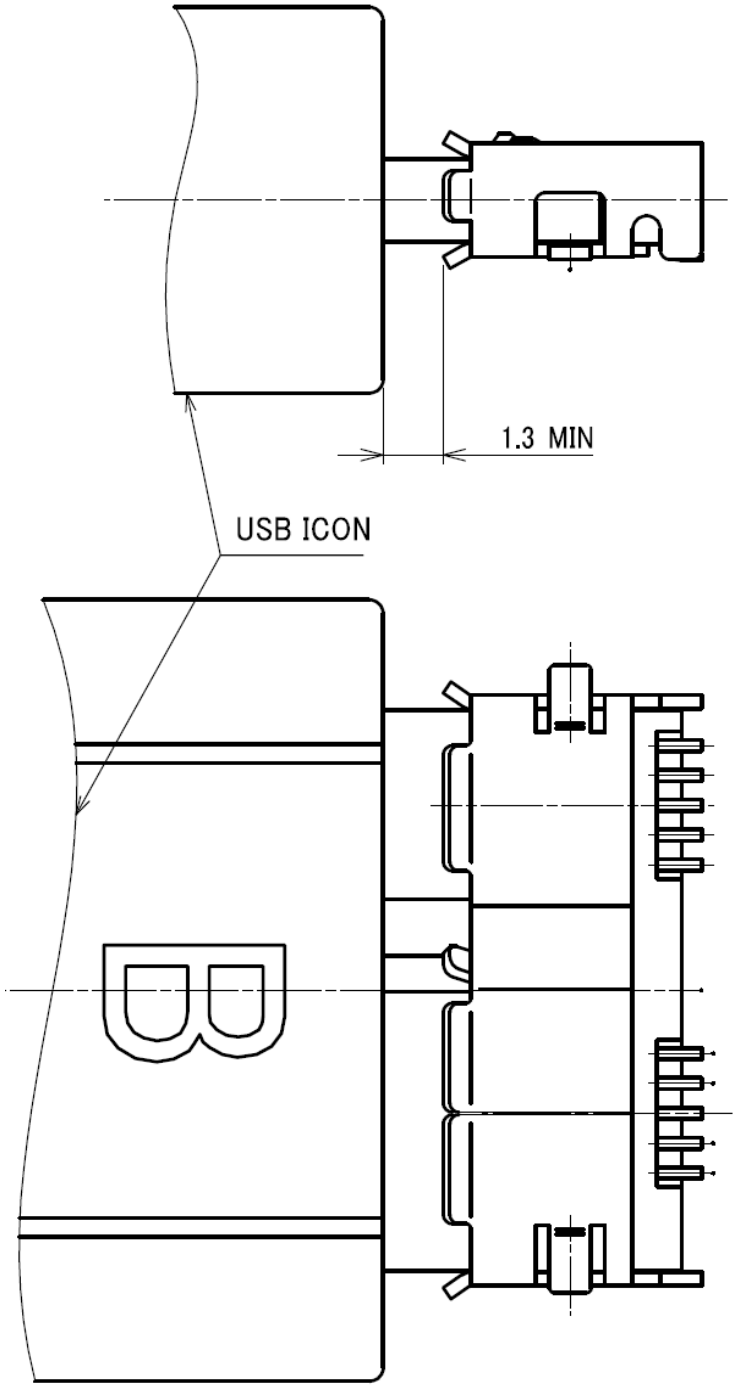


Figure 5-30. Mated USB 3.0 Micro-B Connector

5.8.2 EMI Management

Systems that include USB 3.0 connectors and cable assemblies must meet the relevant EMI/EMC regulations. Because of the complex nature of EMI, it is difficult to specify a component level EMI test for the cable assemblies. However, connector and cable assembly designers, as well as system implementers should pay attention to receptacle and cable plug shielding to ensure a low impedance grounding path. The following are guidelines for EMI management:

- The quality of raw cables should be ensured. The intra-pair skew or the differential to common mode conversion of the SuperSpeed pairs has a significant impact on cable EMI performance and should be controlled within the limits of this specification.
- The cable external braid should be terminated to the cable plug metal shell as close to 360˚ as possible. Without appropriate shielding termination, even a perfect cable with zero intra-pair skew may not meet EMI requirements.
- If not done properly, the wire termination contributes to the differential-to-common-mode conversion. The breakout distance for the wire termination should be kept as small as possible for both EMI and signal integrity. If possible, symmetry should be maintained for the two lines within a differential pair.
- The mating interface between the receptacle and cable plug should have a sufficient number of grounding fingers, or springs to provide a continuous return path from the cable plug to system ground. Friction locks should not compromise ground return connections.
- The receptacle connectors should be designed with a back-shield as part of the receptacle connector metal shell. The back-shield should be designed with a short return path to the system ground plane.
- The receptacle connectors should be connected to metal chassis or enclosures through grounding fingers, screws, or any other way to mitigate EMI.

5.8.3 Stacked Connectors

Stacked USB connectors are commonly used in PC systems. This specification does not explicitly define the stacked USB 3.0 Standard-A receptacles but they are allowed. The following are a few points that should be taken into account when designing a stacked USB 3.0 connector:

- A stacked connector introduces additional crosstalk between the top and bottom connectors. Such crosstalk should be minimized when designing a stacked USB 3.0 connector. The differential NEXT and FEXT should be managed within ~-32 dB (up to the fundamental frequency of 2.5 GHz) between differential pairs in the top and bottom connectors.
- Due to the additional electrical length, the top connector will generally not perform as well as the bottom connector. Connector designers should carefully design the top connector contact geometries and materials to minimize impedance discontinuity. Regardless of how many connectors within a stack one may choose to design, the electrical requirements defined in Section 5.6 must be met.

6 Physical Layer

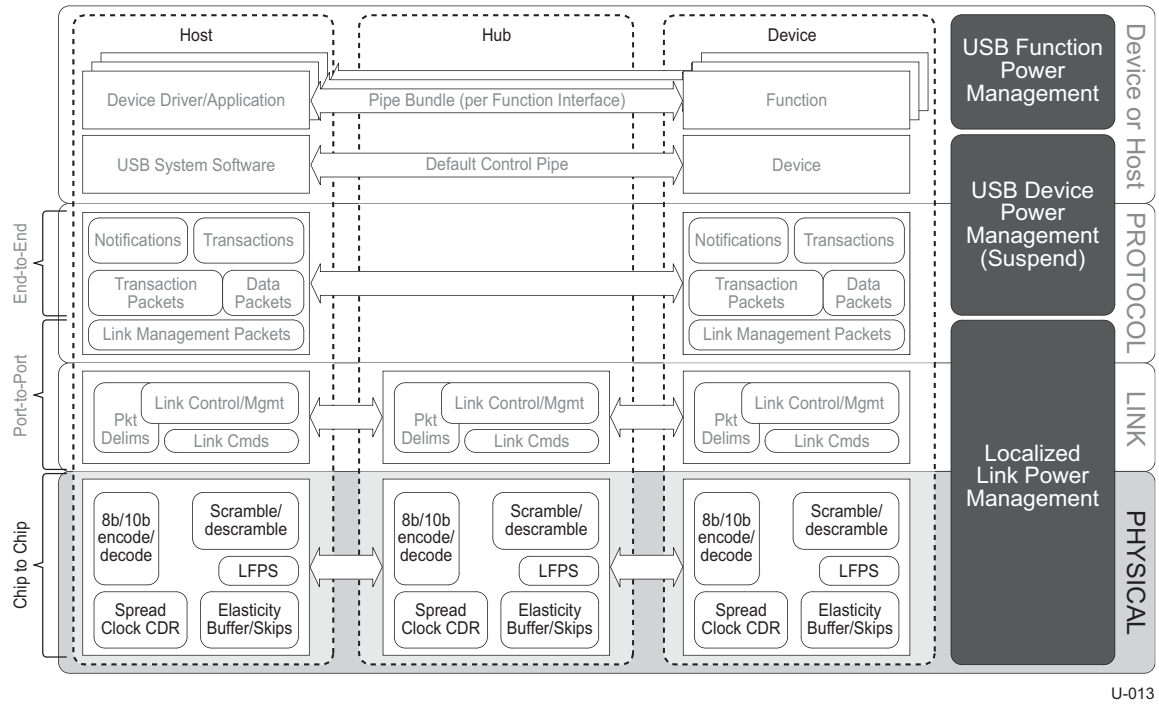


Figure 6-1. SuperSpeed Physical Layer

6.1 Physical Layer Overview

The physical layer defines the signaling technology for the SuperSpeed bus. This chapter defines the electrical requirements of the SuperSpeed physical layer.

This section defines the electrical-layer parameters required for interoperability between SuperSpeed components. Normative specifications are required. Informative specifications may assist product designers and testers in understanding the intended behavior of the SuperSpeed bus.

6.2 Physical Layer Functions

The functions of the physical layer are shown in Figure 6-2, Figure 6-3, and Figure 6-4.

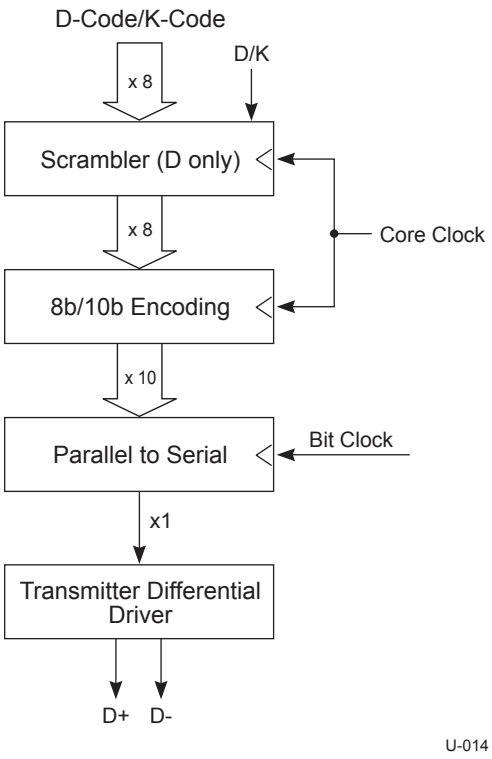


Figure 6-2. Transmitter Block Diagram

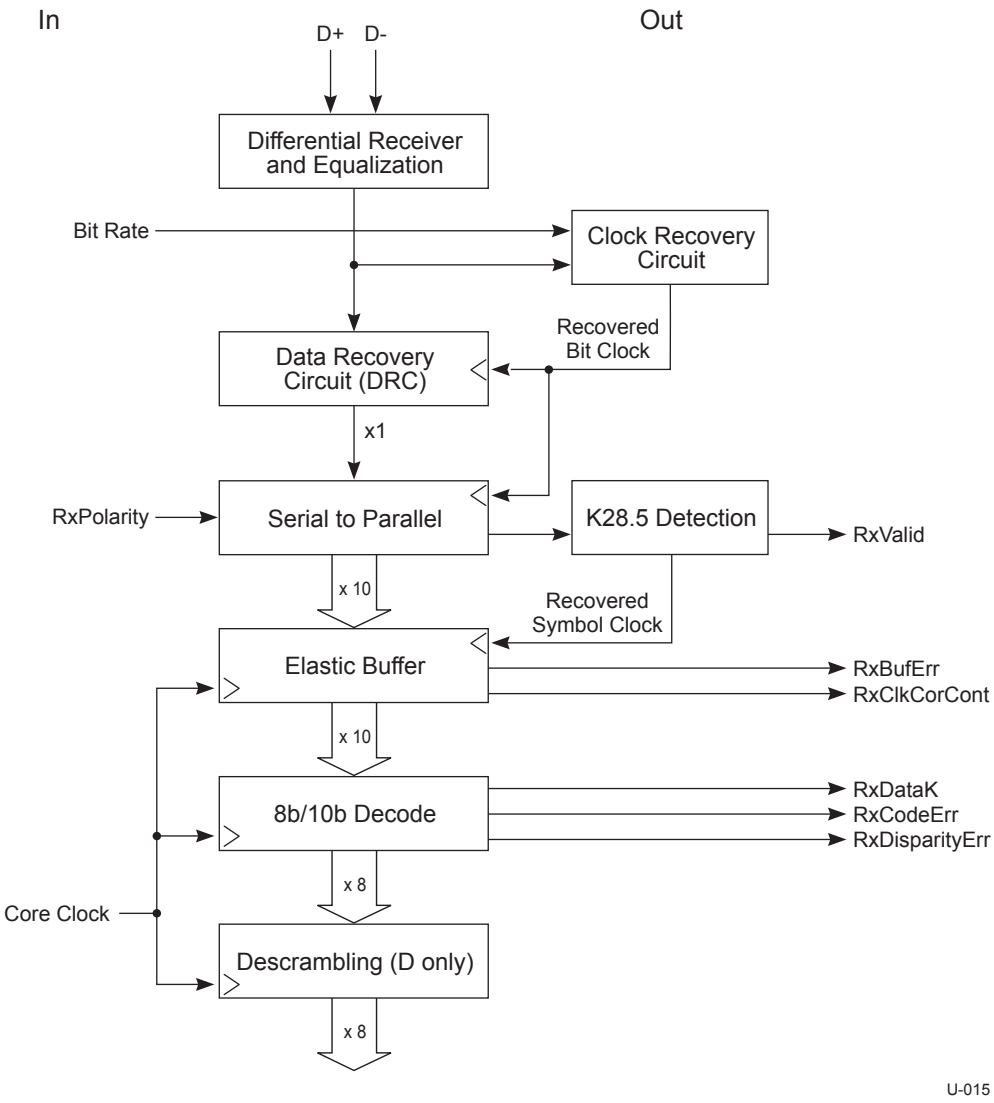


Figure 6-3. Receiver Block Diagram

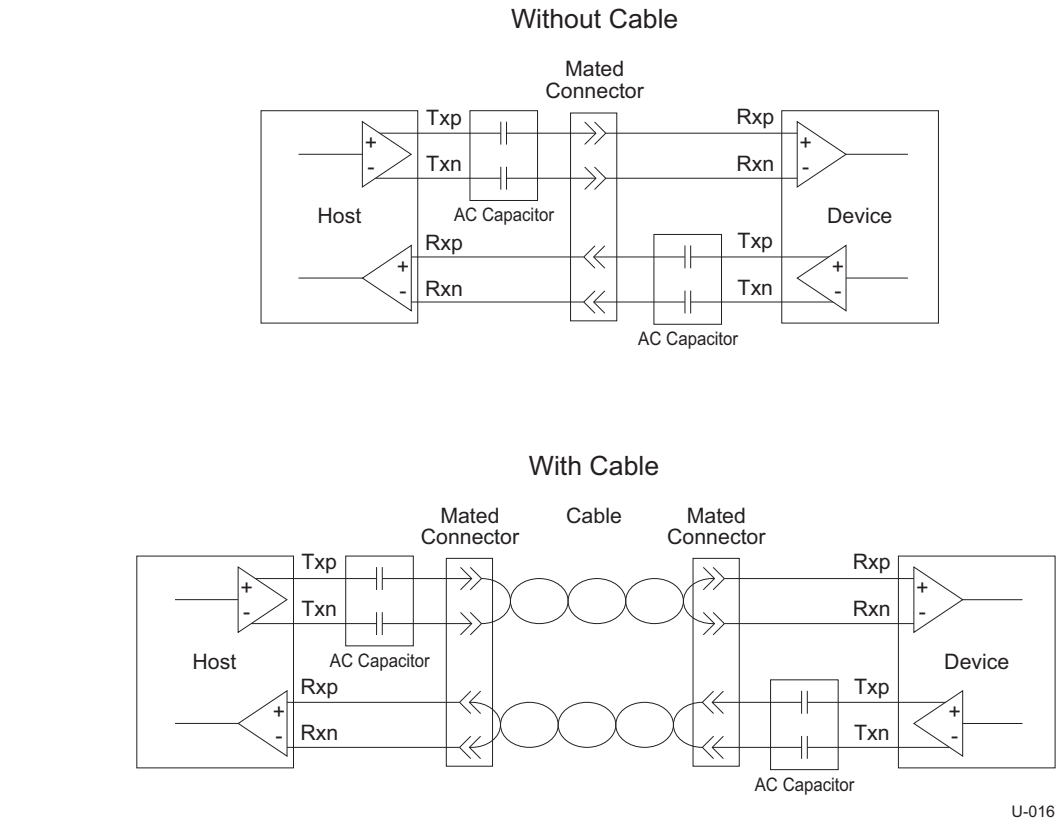


Figure 6-4. Channel Models without a Cable (Top) and with a Cable (Bottom)

6.2.1 Measurement Overview

The normative SuperSpeed eye diagram is to be measured through a compliance channel that represents the sum of a long channel, a short channel, and a 3-meter cable. This requires three separate tests for compliance. These reference channels are described in the *USB SuperSpeed Compliance Methodology* white paper. The eye diagram is measured using the clock recovery function described in Section 6.5.2.

For the long channel case the eye diagram at the receiver is completely closed. An informative receiver equalization function is provided in Section 6.8.2 that is optimized to the compliance channel and is used to open the receiver eye.

This methodology allows a silicon vendor to design the channel and the component as a matched pair. It is expected that a silicon component will have layout guidelines that must be followed in order for the component to meet the overall specification and the eye diagram at the end of the compliance channel.

Note that simultaneous USB 2.0 and SuperSpeed operation is a testing requirement for compliance.

6.2.2 Channel Overview

A PHY is a transmitter and receiver that operate together and are located on the same component. A channel connects two PHYs together with two unidirectional differential pairs of pins for a total of four wires. The PHYs are required to be AC coupled. The AC coupling capacitors are associated with the transmitter.

6.3 Symbol Encoding

SuperSpeed uses the 8b/10b transmission code. The definition of this transmission code is identical to that specified in ANSI X3.230-1994 (also referred to as ANSI INCITS 230-1994), clause 11. As shown in Figure 6-5, ABCDE maps to abcdei and FGH maps to fghj.

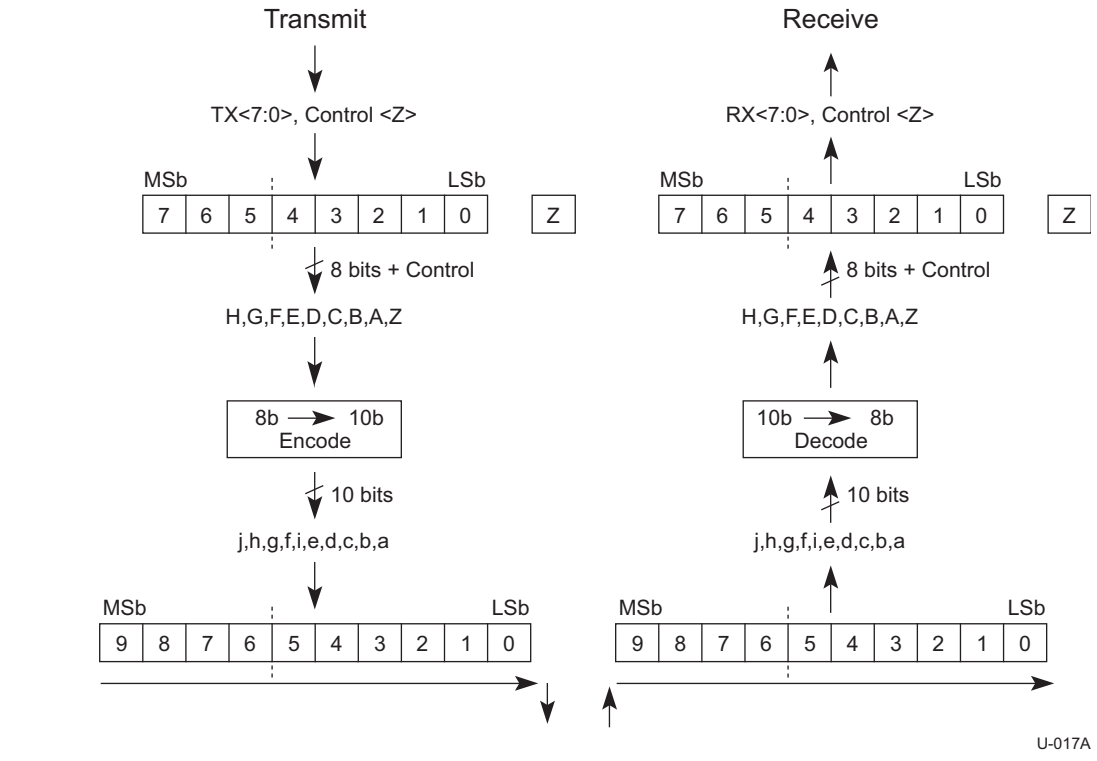


Figure 6-5. Character to Symbol Mapping

6.3.1 Serialization and Deserialization of Data

The bits of a Symbol are placed starting with bit "a" and ending with bit "j." This is shown in Figure 6-6.

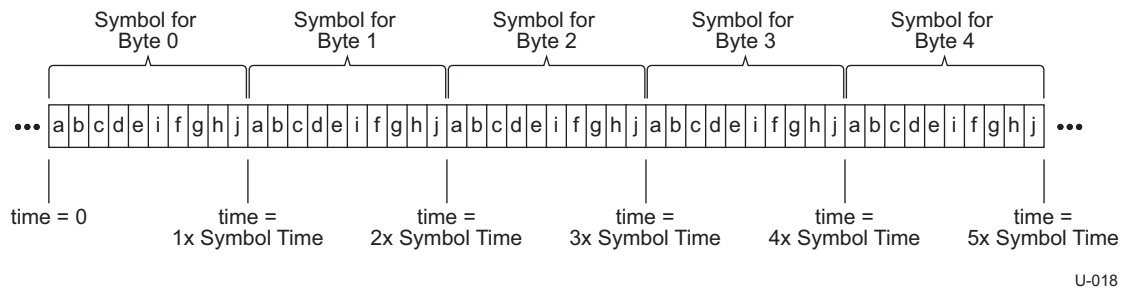


Figure 6-6. Bit Transmission Order

6.3.2 Normative 8b/10b Decode Rules

1. A Transmitter is permitted to pick any disparity when first transmitting differential data after being in an Electrical Idle state. The Transmitter shall then follow proper 8b/10b encoding rules until the next Electrical Idle state is entered.
2. The initial disparity for a Receiver is the disparity of the first Symbol used to obtain Symbol lock.
3. Disparity may also be-reinitialized if Symbol lock is lost and regained during the transmission of differential information due to a burst error event.
4. All following received Symbols after the initial disparity is set shall be in the proper column corresponding to the current running disparity.
5. Receive disparity errors do not directly cause the link to retrain.
6. If a disparity error or 8b/10 Decode error is detected, the physical layer shall inform the link layer.

6.3.3 Data Scrambling

The scrambling function is implemented using a free running Linear Feedback Shift Register (LFSR). On the Transmit side, scrambling is applied to characters prior to the 8b/10b encoding. On the receive side, descrambling is applied to characters after 8b/10b decoding. The LFSR is reset whenever a COM symbol is sent or received.

The LFSR is graphically represented in Figure 6-7. Scrambling or unscrambling is performed by serially XORing the 8-bit (D0-D7) character with the 16-bit (D0-D15) output of the LFSR. An output of the LFSR, D15, is XORed with D0 of the data to be processed. The LFSR and data register are then serially advanced and the output processing is repeated for D1 through D7. The LFSR is advanced after the data is XORed.

The mechanism to notify the physical layer to disable scrambling is implementation specific and beyond the scope of this specification.

The data scrambling rules are as follows:

1. The LFSR implements the polynomial: $G(X)=X^{16}+X^5+X^4+X^3+1$
2. The LFSR value shall be advanced eight serial shifts for each Symbol except for SKP.
3. All 8b/10b D-codes, except those within the Training Sequence Ordered Sets shall be scrambled.
4. K codes shall not be scrambled.
5. The initialized value of an LFSR seed (D0-D15) shall be FFFFh. After COM leaves the Transmitter LFSR, the LFSR on the transmit side shall be initialized. Every time COM enters the Receive LFSR, the LFSR on the receive side shall be initialized. This also applies to the BRST sequence during loopback mode (see section 6.8.4.1).

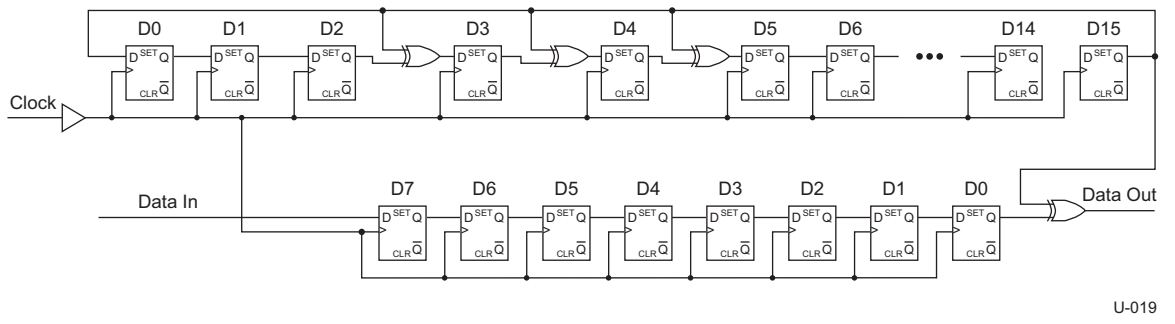


Figure 6-7. LFSR with Scrambling Polynomial

IMPLEMENTATION NOTE

Disabling Scrambling

Disabling scrambling is intended to help simplify test and debug equipment. Control of the exact data patterns is useful in a test and debug environment. Since scrambling is reset at the physical layer, there is no reasonable way to reliably control the state of the data transitions through software. The Disable Scrambling bit is provided in the training sequence for this purpose.

The mechanism(s) and/or interface(s) used to notify the physical layer to disable scrambling is component implementation specific and beyond the scope of this specification.

For more information on scrambling, refer to Appendix B.

6.3.4 8b/10b Decode Errors

An 8b/10b Decode error shall occur when a received Symbol does not match any of the valid 8b/10b Symbols listed in Appendix A. Any received 8b/10b Symbol that does not match any of the valid 8b/10b Symbols listed in Appendix A shall be forwarded to the link layer by substituting a K28.4 symbol (refer to Table 6-1). 8b/10b errors may not directly initiate Recovery.

6.3.5 Special Symbols for Framing and Link Management

The 8b/10b encoding scheme provides Special Symbols that are distinct from the Data Symbols used to represent characters. These Special Symbols are used for various Link Management mechanisms described later. Table 6-1 lists the Special Symbols used and provides a brief description for each. Special Symbols must follow the proper 8b/10b disparity rules. The compliance tests are defined in the *USB SuperSpeed Compliance Methodology* white paper.

Table 6-1. Special Symbols

Encoding	Symbol	Name	Description
K28.1	SKP	Skip	Compensates for different bit rates between two communicating ports. SKPs may be dynamically inserted or removed from the data stream.
K28.2	SDP	Start Data Packet	Marks the start of a Data Packet Payload
K28.3	EDB	End Bad	Marks the end of a nullified Packet
K28.4	SUB	Decode Error Substitution	Symbol substituted by the 8b/10b decoder when a Decode error is detected.
K28.5	COM	Comma	Used for symbol alignment
K28.6	-----	-----	Reserved
K27.7	SHP	Start Header Packet	Marks the start of a Data Packet, Transaction Packet or Link Management Packet
K29.7	END	End	Marks the end of a packet
K30.7	SLC	Start Link Command	Marks the start of a Link Command
K23.7	EPF	End Packet Framing	Marks the end of a packet framing

6.4 Link Initialization and Training

This section defines the sequences that are used for configuration and initialization. The sequences are used by the Initialization State Machine (refer to Chapter 7) for the following functions:

- Configuring and initializing the link
- Bit-lock and symbol lock
- Rx equalization training
- Lane polarity inversion

Training sequences are composed of Ordered Sets used for initializing bit alignment, Symbol alignment and optimizing the equalization. Training sequence Ordered Sets are never scrambled but are always 8b/10b encoded.

Bit lock refers to the ability of the Clock/Data Recovery (CDR) circuit to extract the phase and frequency information from the incoming data stream. Bit lock is accomplished by sending a sufficiently long sequence of bits (D10.2 symbol containing alternating 0s and 1s) so the CDR roughly centers the clock within the bit.

Once the CDR is properly recovering data bits, the next step is to locate the start and end of a 10-bit symbol. For this purpose, the special K-Code called COMMA is selected from the 8b/10b codes. The bit pattern of the COMMA code is unique, so that it is never found in other data patterns,

including any combination of a D-Code appended to any other D-Code or appended to any K-Code. This applies to any polarity of code. The only exception is for various bit patterns that include a bit error.

Training sequences (TS1 or TS2) are transmitted consecutively and can only be interrupted by SKP Ordered Sets occurring between Ordered Sets (between consecutive TS1 sets, consecutive TS2 sets, or when TS1 is followed by TS2).

6.4.1 Normative Training Sequence Rules

Training sequences are composed of Ordered Sets used for initializing bit alignment, symbol alignment, and receiver equalization.

The following rules apply to the training sequences:

- Training sequence Ordered Sets shall be 8b/10b encoded.
- Transmission of a TS1 or TS2 Ordered Set shall not be interrupted by SKP Ordered Sets. SKP Ordered Sets shall be inserted before, or after, completion of any TS1 or TS2 Ordered Set.
- No SKP Ordered Sets are to be transmitted during the entire TSEQ time (65,536 ordered sets). This means that the PHY must manage elasticity buffer differently than during normal operation.

Additional rules for the use of TSEQ, TS1, and TS2 Ordered Sets can be found in Chapter 7.

6.4.1.1 Training Control Bits

The training control bits are found in the Link Functionality symbol within the TS1 and TS2 ordered sets. They are described in Table 6-5.

Bit 0 and bit 2 of the link configuration field shall not be set to 1 simultaneously. If a receiver detects this condition in the received Link configuration field, then all of the training control bits shall be ignored.

6.4.1.2 Training Sequence Values

The TSEQ training sequence repeats 65,536 times to allow for testing many coefficient settings.

Table 6-2. TSEQ Ordered Set

Symbol Number	Name	Value
0	K28.5	COM (Comma)
1	D31.7	0xFF
2	D23.0	0x17
3	D0.6	0xC0
4	D20.0	0x14
5	D18.5	0xB2
6	D7.7	0xE7
7	D2.0	0x02
8	D2.4	0x82
9	D18.3	0x72
10	D14.3	0x6E

Symbol Number	Name	Value
11	D8.1	0x28
12	D6.5	0xA6
13	D30.5	0xBE
14	D13.3	0x6D
15	D31.5	0xBF
16-31	D10.2	0x4A

Table 6-3. TS1 Ordered Set

Symbol Number	Encoded Values	Description
0-3	K28.5	COM (Comma)
4	D0.0	Reserved for future use
5	See Table 6-5	Link Functionality
6-15	D10.2	TS1 Identifier

Table 6-4. TS2 Ordered Set

Symbol Number	Encoded Values	Description
0-3	K28.5	COM (Comma)
4	D0.0	Reserved
5	See Table 6-5	Link Functionality
6-15	D5.2	TS2 Identifier

Table 6-5. Link Configuration Field

Bit	TS1 Symbol 5	Description
Bit 0	0 = Normal Training 1 = Reset	Reset is set by the Host only in order to reset the device.
Bit 1	Set to 0	Reserved for future use.
Bit 2	0 = Loopback de-asserted 1 = Loopback asserted	When set, the receiving component enters digital loopback.
Bit 3	0 = Disable Scrambling de-asserted 1 = Disable Scrambling asserted	When set, the receiving component disables scrambling.
Bit 4:7	Set to 0	Reserved for future use.

6.4.2 Lane Polarity Inversion

During the TSEQ training sequence, the Receiver must use the D10.2 Symbol within the TSEQ Ordered Set to determine lane polarity inversion (Rxp and Rxn are swapped). If polarity inversion has occurred, the D10.2 symbols within the TSEQ ordered set will be received as D21.5 instead of D10.2 and the receiver must invert the polarity of the received bits. This must be done before the TSEQ symbols 1-15 are used since these symbols are not all symmetric under inversion in the 8b/10b domain. If the receiver does not use the TSEQ training sequence, then the polarity inversion may be checked against the D10.2 symbol in the TS1 ordered set.

6.4.3 Elasticity Buffer and SKP Ordered Set

The SuperSpeed architecture supports a separate reference clock source on each side of the SuperSpeed link. The accuracy of each reference clock is required to be within +-300 ppm. This gives a maximum frequency difference between the two devices of the link of +- 600 ppm. In addition, SSC creates a frequency delta that has a maximum difference of 5000 ppm. The total magnitude of the frequency delta can range from -5300 to 300 ppm. This frequency delta is managed by an elasticity buffer that consumes or inserts SKP ordered sets.

SKP Ordered Sets shall be used to compensate for frequency differences between the two ends of the link. The transmitter sends SKP ordered sets at an average of every 354 symbols. However, SKP ordered sets shall not be inserted within any packet. The transmitter is allowed to buffer the SKP ordered sets up to a maximum of four SKP ordered sets. The receiver must implement an elasticity buffer capable of buffering (or starving) eight symbols of data.

SKP Rules (Host/Device/Hub):

- The SKP Ordered Set shall consist of a SKP K-Symbol followed by a SKP K-Symbol. A SKP Ordered Set represents two Symbols that can be used for clock compensation.
- A device must keep a running count of the number of transmitted symbols since the last SKP Ordered Set. The value of this count will be referred to as Y. The value of Y is reset whenever the transmitter enters Polling.Active or exits U1/U2/U3 low power states.
- Unless otherwise specified, a transmitter shall insert the integer result of Y/354 calculation Ordered Sets immediately after each transmitted TS1 Ordered Set, TS2 Ordered Set, LMP, TP, DP, or Logical Idle. During training only, a transmitter is allowed the option of waiting to insert 2 SKP Ordered Sets when the integer result of Y/354 reaches 2. A transmitter shall not transmit SKP Ordered Sets at any other time.

  Note: The non-integer remainder of the Y/354 SKP calculation shall not be discarded and shall be used in the calculation to schedule the next SKP Ordered Set.
- SKP Commands do not count as interruptions when monitoring for Ordered Sets (i.e., consecutive TS1, TS2 Ordered Sets in Polling and Recovery).

Table 6-6. SKP Ordered Set Structure

Symbol Number	Encoded Values	Description
0	K28.1	SKP
1	K28.1	SKP

6.4.4 Compliance Pattern

Entry to the Compliance Mode is described in Chapter 7. This initiates the transmission of the pseudo-random data pattern generated by the scrambled D0.0 compliance sequence. SKPs are not sent during the compliance pattern. The compliance pattern shall be transmitted continuously or until a ping LFPS (refer to Section 6.9) is detected at the receiver. Upon detection of a ping LFPS, the compliance pattern shall advance to the next compliance pattern. Upon detection of a reset, the compliance pattern shall be terminated. The compliance pattern sequences are described in Table 6-7.

Table 6-7. Compliance Pattern Sequences

Compliance Pattern	Value	Description
CP0	D0.0 scrambled	A pseudo-random data pattern that is exactly the same as logical idle (refer to Chapter 7) but does not include SKP sequences
CP1	D10.2	Nyquist frequency
CP2	D24.3	Nyquist/2
CP3	K28.5	COM pattern
CP4	LFPS	The low frequency periodic signaling pattern
CP5	K28.7	With de-emphasis
CP6	K28.7	Without de-emphasis
CP7	50-250 1's and 0's	With de-emphasis. Repeating 50-250 1's and then 50-250 0's.
CP8	50-250 1's and 0's	With without de-emphasis. Repeating 50-250 1's and then 50-250 0's.

Note: Unless otherwise noted, scrambling is disabled for compliance patterns.

6.5 Clock and Jitter

6.5.1 Informative Jitter Budgeting

The jitter for USB 3.0 is budgeted among the components that comprise the end to end connections: the transmitter, channel (including packaging, connectors, and cables), and the receiver. The jitter budget is derived at the silicon pads. The Dj distribution is the dual Dirac method. Table 6-8 lists Tx, Rx, and channel jitter budgets.

Table 6-8. Informative Jitter Budgeting at the Silicon Pads

Jitter Contribution (ps)	Rj[1,2]	Dj[3]	Tj[4] at 10^{-12}
Tx[6]	2.42	41	75
Media[5]	2.13	45	75
Rx	2.42	57	91
Total:	4.03	143	200

Notes:

1. Rj is the sigma value assuming a Gaussian distribution.
2. Rj Total is computed as the Root Sum Square of the individual Rj components.
3. Dj budget is using the Dual Dirac method.
4. Tj at a 10^{-12} BER is calculated as 14.068 * Rj + Dj.
5. The media budget includes the cancellation of ISI from the appropriate Rx equalization function.
6. Tx is measured after application of the JTF.

Note: Captive cables must meet the mated connector requirements specified in Section 5.6.1.2. But a captive cable is not considered a stand-alone component. For electrical budgeting purposes, a captive cable is considered to be part of a device, and must meet the device jitter requirements listed in Table 6-8.

6.5.2 Normative Clock Recovery Function

The Tx Phase jitter measurement is performed using a standard clock recovery, shown in Figure 6-8. For information on the golden PLL measurement refer to the latest version of INCITS TR-35-2004, *INCITS Technical Report for Information Technology – Fibre Channel – Methodologies for Jitter and Signal Quality Specification (FC-MJSQ)*.

The clock recovery function is given by Equations 1-3. A schematic of the general clock recovery function is shown in Figure 6-8. As shown, the clock recovery circuit has a low pass response. After the recovered clock is compared (subtracted) to the data, the overall clock recovery becomes a high pass function. This is shown in Figure 6-9.

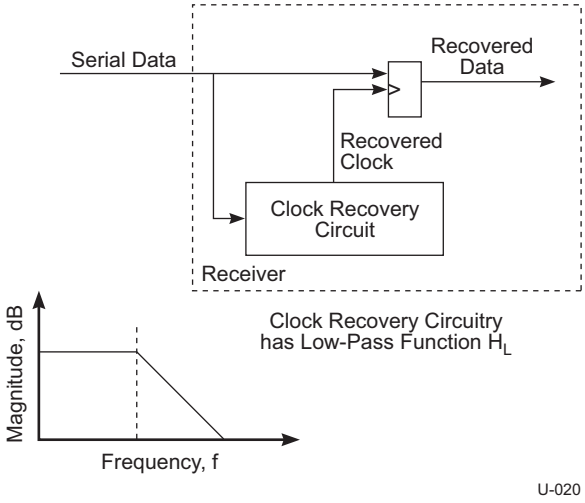


Figure 6-8. Jitter Filtering – "Golden PLL" and Jitter Transfer Functions

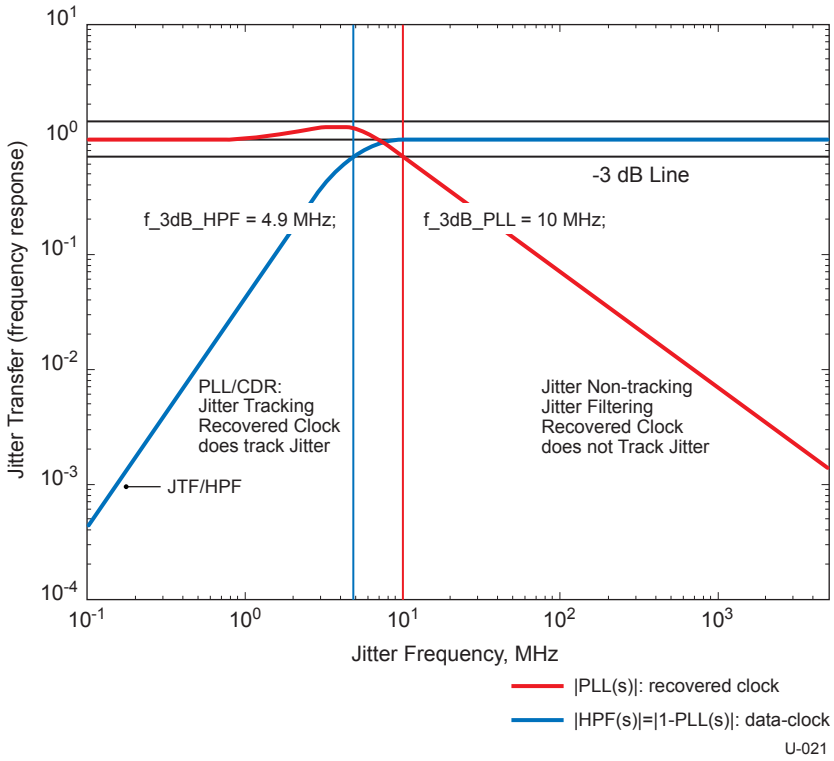


Figure 6-9. "Golden PLL" and Jitter Transfer Functions

The equations for these functions are:

(1) $$H_{CDR}(s) = \frac{2s\zeta\omega_n + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2}$$

and

(2) $$JTF(s) = \frac{s^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

where ω_n is the natural frequency and ξ is the damping factor. The relationship to the 3 dB frequency is

(3) $$\omega_{3dB} = \omega_n\left(1 + 2\zeta^2 + \left[\left(1 + 2\zeta^2\right)^2 + 1\right]^{\frac{1}{2}}\right)^{\frac{1}{2}}$$

As shown in Figure 5-7, the corner frequency $\omega_{3dB} = 2\pi 10^7$ and $\zeta = 0.707$. This transfer function has a maximum peaking of 2 dB.

6.5.3 Normative Spread Spectrum Clocking (SSC)

All ports are required to have Spread Spectrum Clocking (SSC) modulation. Providing the same SSC clock to two different components is allowed but not required, the SSC can be generated asynchronously. The SSC profile is not specified and is vendor specific. The SSC modulation requirement is listed in Table 6-9. The SSC modulation may not violate the phase slew rate described in Section 6.5.4.

Table 6-9. SSC Parameters

Symbol	Description	Limits		Units	Note
		Min	Max		
$t_{SSC\text{-}MOD\text{-}RATE}$	Modulation Rate	30	33	kHz	
$t_{SSC\text{-}FREQ\text{-}DEVIATION}$	SSC deviation	+0/-4000	+0/-5000	ppm	1, 2

Note:

1. The data rate is modulated from 0 ppm to -5000 ppm of the nominal data rate frequency and scales with data rate.
2. This is measured below 2 MHz only.

An example of the period modulation from triangular SSC is shown in Figure 6-10.

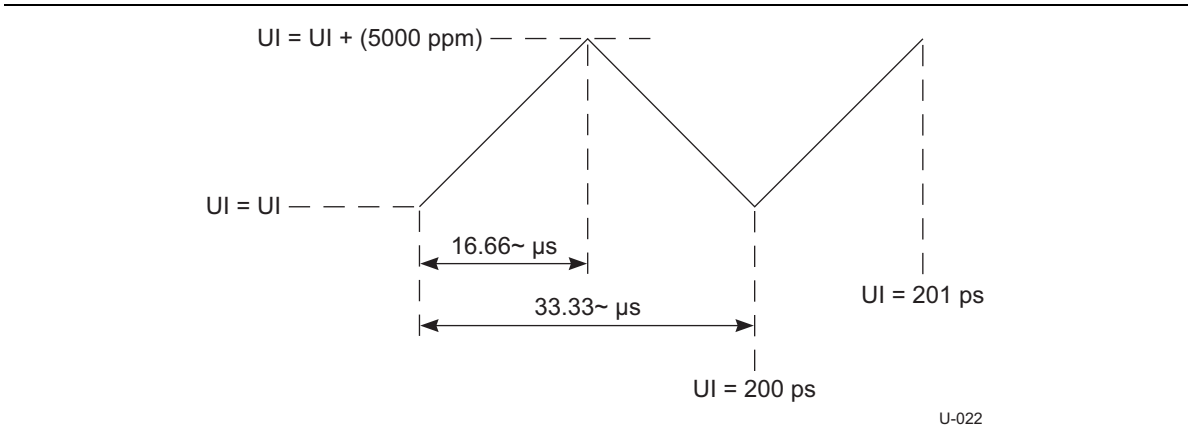


Figure 6-10. Period Modulation from Triangular SSC

6.5.4 Normative Slew Rate Limit

The CDR is a slew rate limited phase tracking device. The combination of SSC and all other jitter sources within the bandwidth of the CDR must not exceed the maximum allowed slew rate.

This measurement is performed by filtering the phase jitter with the CDR transfer function and taking the first difference of the phase jitter to obtain the filtered period jitter. The peak of the period jitter must not exceed $T_{CDR\_SLEW\_MAX}$ listed in Table 6-10.

Additional details on the slew rate measurement are available in *USB 3.0 Jitter Budgeting*.

6.6 Signaling

6.6.1 Eye Diagrams

The eye diagrams are a graphical representation of the voltage and time limits of the signal. This eye mask applies to jitter after the application of the appropriate jitter transfer function and reference receiver equalization. In all cases, the eye is to be measured for 10^6 consecutive UI. The budget for the link is derived assuming a total 10^{-12} bit error rate and is extrapolated to a measurement of 10^6 UI assuming the random jitter is Gaussian.

Figure 6-11 shows the eye mask used for all eye diagram measurements. Referring to the figure, the time is measured from the crossing points of Txp/Txn. The time is called the eye width, and the voltage is the eye height. The eye height is to be measured at the maximum opening (at the center of the eye width ± 0.05 UI).

The eye diagrams are to be centered using the jitter transfer function (JTF). The recovered clock is obtained from the data and processed by the JTF. The center of the recovered clock is used to position the center of the data in the eye diagram.

The eye diagrams are to be measured into 50-Ω single-ended loads.

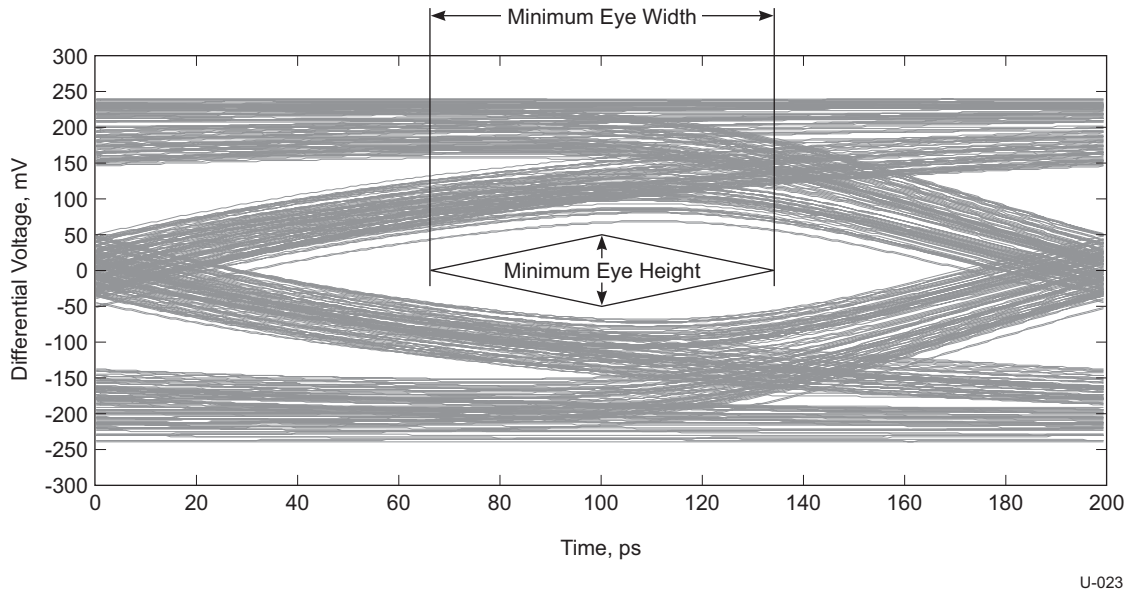


Figure 6-11. Generic Eye Mask

6.6.2 Voltage Level Definitions

Referring to Figure 6-12, the differential voltage, V_{DIFF}, is the voltage on Txp (Rxp at the receiver) with respect to Txn (Rxn at the receiver). V_{DIFF} is the same voltage as the swing on the single signal of one conductor. The differential voltage is

(4) $V_{DIFF} = Txp - Txn$

The total differential voltage swing is the peak to peak differential voltage, $V_{DIFF-PP}$. This is twice the differential voltage. The peak to peak differential voltage is

(5) $V_{DIFF-PP}=2 * V_{DIFF}$

The Common Mode Voltage (V_{CM}) is the average voltage present on the same differential pair with respect to ground. This is measured, with respect to ground, as

(6) $V_{CM} = (Txp + Txn) / 2$

DC is defined as all frequency components below F_{DC} = 30 kHz. AC is defined as all frequency components at or above F_{DC} = 30 kHz. These definitions pertain to all voltage and current specifications.

An example waveform is shown in Figure 6-12. In this waveform, the peak-to-peak differential voltage, $V_{DIFF-PP}$ is 800 mV. The differential voltage, V_{DIFF}, is 400 mVPP. Note that while the center crossing point for both Txp and Txn is shown at 300 mV, the corresponding crossover point for the differential voltage is at 0.0 V. The center crossing point at 300 mV is also the common mode voltage, V_{CM}. Note these waveforms include de-emphasis. The actual amount of de-emphasis can vary depending on the transmitter setting according to the allowed ranges in Table 6-10.

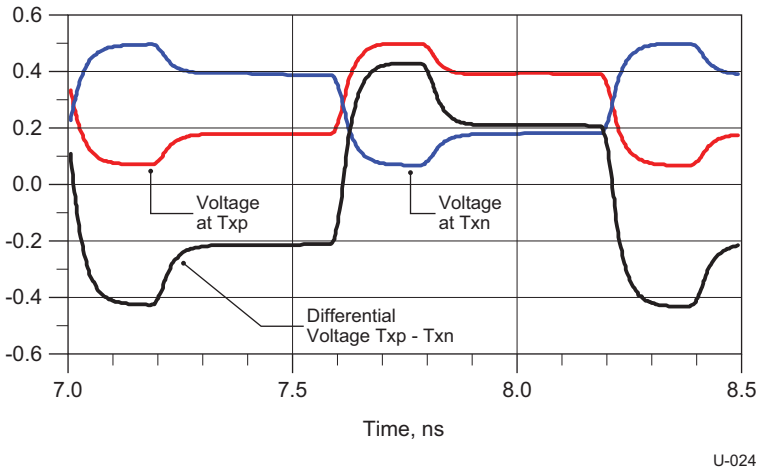


Figure 6-12. Single-ended and Differential Voltage Levels

6.6.3 Tx and Rx Input Parasitics

Tx and Rx input parasitics are specified by the lumped circuit shown in Figure 6-13.

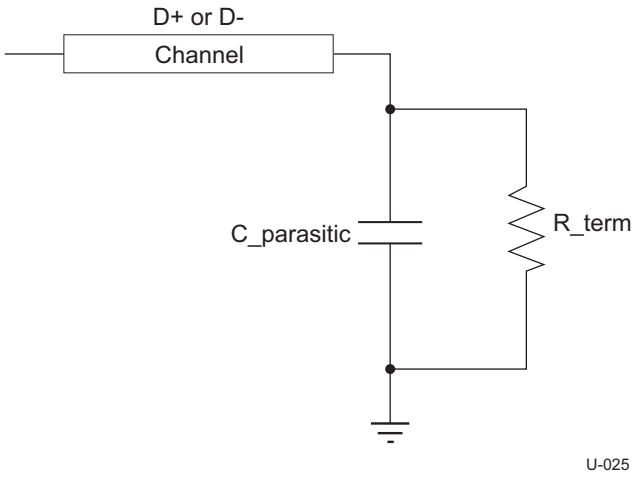


Figure 6-13. Device Termination Schematic

In this circuit, the input buffer is simplified to a termination resistance in parallel with a parasitic capacitor. This simplified circuit is the load impedance.

6.7 Transmitter Specifications

6.7.1 Transmitter Electrical Parameters

Peak (p) and peak-peak (p-p) are defined in Section 6.6.2.

Table 6-10. Transmitter Normative Electrical Parameters

Symbol	Parameter	5.0 GT/s	Units	Comments
UI	Unit Interval	199.94 (min) 200.06 (max)	ps	The specified UI is equivalent to a tolerance of ±300 ppm for each device. Period does not account for SSC induced variations.
$V_{TX-DIFF-PP}$	Differential p-p Tx voltage swing	0.8 (min) 1.2 (max)	V	Nominal is 1 V p-p
$V_{TX-DIFF-PP-LOW}$	Low-Power Differential p-p Tx voltage swing	0.4 (min) 1.2 (max)	V	Refer to Section 6.7.2. There is no de-emphasis requirement in this mode. De-emphasis is implementation specific for this mode.
$V_{TX-DE-RATIO}$	Tx de-emphasis	3.0 (min) 4.0 (max)	dB	Nominal is 3.5 dB
$R_{TX-DIFF-DC}$	DC differential impedance	72 (min) 120 (max)	Ω	
$V_{TX-RCV-DETECT}$	The amount of voltage change allowed during Receiver Detection	0.6 (max)	V	Detect voltage transition should be an increase in voltage on the pin looking at the detect signal to avoid a high impedance requirement when an “off” receiver’s input goes below ground.
$C_{AC-COUPLING}$	AC Coupling Capacitor	75 (min) 200 (max)	nF	All Transmitters shall be AC coupled. The AC coupling is required either within the media or within the transmitting component itself.
$t_{CDR\_SLEW\_MAX}$	Maximum slew rate	10	ms/s	See the jitter white paper for details on this measurement. This is a df/ft specification; refer to Section 6.5.4 for details.

The values in Table 6-11 are informative and not normative. They are included in this document to provide some guidance beyond the normative requirements in Table 6-10 for transmitter design and development. A transmitter can be fully compliant with the normative requirements of the specification and not meet all the values in this table (many of which are immeasurable in a finished product). Similarly, a transmitter that meets all the values in this table is not guaranteed to be in full compliance with the normative part of this specification.

Table 6-11. Transmitter Informative Electrical Parameters at Silicon Pads

Symbol	Parameter	5.0 GT/s	Units	Comments
$t_{MIN-PULSE-Dj}$	Deterministic min pulse	0.96	UI	Tx pulse width variation that is deterministic
$t_{MIN-PULSE-Tj}$	Tx min pulse	0.90	UI	Min Tx pulse at 10^{-12} including Dj and Rj
t_{TX-EYE}	Transmitter Eye	0.625 (min)	UI	Includes all jitter sources
$t_{TX-DJ-DD}$	Tx deterministic jitter	0.205 (max)	UI	Deterministic jitter only assuming the Dual Dirac distribution
$C_{TX-PARASITIC}$	Tx input capacitance for return loss	1.25 (max)	pf	Parasitic capacitance to ground
R_{TX-DC}	Transmitter DC common mode impedance	18 (min) 30 (max)	Ω	DC impedance limits to guarantee Receiver detect behavior. Measured with respect to AC ground over a voltage of 0-500 mV.
$I_{TX-SHORT}$	Transmitter short-circuit current limit	60 (max)	mA	The total current Transmitter can supply when shorted to ground.
$V_{TX-DC-CM}$	Transmitter DC common-mode voltage	0 (min) 2.2 (max)	V	The instantaneous allowed DC common-mode voltages at the connector side of the AC coupling capacitors.
$V_{TX-CM-AC-PP\_ACTIVE}$	Tx AC common mode voltage active	100 mV	mVp-p	Maximum mismatch from Txp + Txn for both time and amplitude.
$V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$	Absolute DC Common Mode Voltage between U1 and U0	200 (max)	mV	
$V_{TX-IDLE-DIFF-AC-pp}$	Electrical Idle Differential Peak – Peak Output Voltage	0 (min) 10 (max)	mV	
$V_{TX-IDLE-DIFF-DC}$	DC Electrical Idle Differential Output Voltage	0 (min) 10 (max)	mV	Voltage must be low pass filtered to remove any AC component. This limits the common mode error when resuming U1 to U0.

6.7.2 Low Power Transmitter

In addition to the full swing transmitter specification, an optional low power swing transmitter is also specified for SuperSpeed applications. A low power swing transmitter is typically used in systems that are sensitive to power and noise interference, and have a relative short channel. The requirement as to whether a transmitter needs to support full swing, low power swing, or both swings, is dependent on its usage model. All SuperSpeed transmitters must support full swing, while support for low power swing is optional. The method by which the output swing is selected is not defined in the specification, and is implementation specific.

While two different transmitters are specified, only a single receiver specification is defined. This implies that receiver margins (as specified in Table 6-13) must be met if a low power transmitter is used.

6.7.3 Transmitter Eye

The eye mask is measured using the compliance data patterns (CP0 for DJ and CP1 for RJ) as described in Section 6.4.4. Eye height is measured for 10^6 consecutive UI. Jitter is extrapolated from 10^6 UI to 10^{-12} BER,

Table 6-12. Normative Transmitter Eye Mask at Test Point TP1

Signal Characteristic	Minimal	Nominal	Maximum	Units	Note
Eye Height	100		1200	mV	2, 4
Dj			0.43	UI	1,2,3
Rj			0.23	UI	1,2,3, 5
Tj			0.66	UI	1,2,3

Notes:

1. Measured over 10^6 consecutive UI and extrapolated to 10^{-12} BER.
2. Measured after receiver equalization function.
3. Measured at end of reference channel and cables at TP1 in Figure 6-14.
4. The eye height is to be measured at the maximum opening (at the center of the eye width ± 0.05 UI).
5. The Rj specification is calculated as 14.069 times the RMS random jitter for 10^{-12} BER.

The compliance testing setup is shown in Figure 6-14. All measurements are made at the test point (TP1), and the Tx specifications are applied after processing the measured data with the compliance reference equalizer transfer function described in the next section.

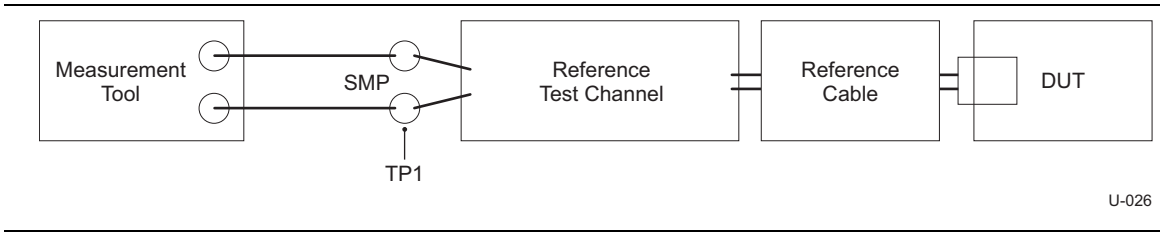


Figure 6-14. Tx Normative Setup with Reference Channel

6.7.4 Tx Compliance Reference Receiver Equalize Function

The normative transmitter eye is captured at the end of the reference channel. At this point the eye may be closed. To open the eye so it can be measured a reference Rx equalizer, is applied to the signal. Details of the reference equalizer are contained in Section 6.8.2.

6.7.5 Informative Transmitter De-emphasis

The channel budgets and eye diagrams were derived using a $V_{TX\text{-}DE\text{-}RATIO}$ of transmit de-emphasis for both the Host and the Device reference channels. An example differential peak-to-peak de-emphasis waveform is shown in Figure 6-15.

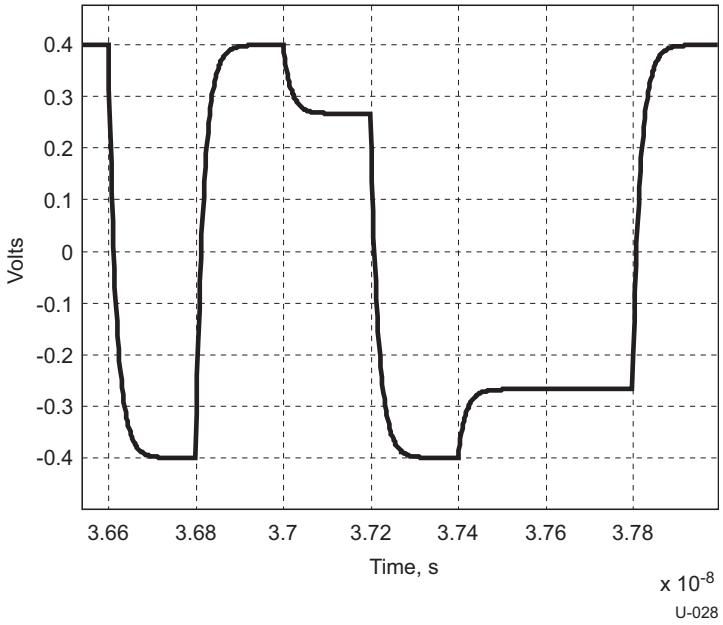


Figure 6-15. De-Emphasis Waveform

6.7.6 Entry into Electrical Idle, U1

Electrical Idle is a steady state condition where the Transmitter Txp and Txn voltages are held constant at the same value and the Receiver Termination is within the range specified by $Z_{RX\text{-}DC}$. Electrical Idle is used in the power saving state of U1.

The low impedance common mode and differential Receiver terminations values (see Table 6-13) must be met in Electrical Idle. The Transmitter can be in either a low or high impedance mode during Electrical Idle.

6.8 Receiver Specifications

6.8.1 Receiver Equalization Training

The receiver equalization training sequence, detailed in Section 6.3.1, can be used to train the receiver equalizer. The TSEQ training sequence is designed to provide a spectrally rich data pattern that is useful for training typical receiver equalization architectures. In addition, a high edge density pattern is interleaved with the data to help the CDR maintain bit lock. The TSEQ training sequence repeats 65536 times to allow for testing many coefficient settings. No SKPs are inserted during the TSEQ training sequence. The frequency spectrum of the TSEQ sequence is shown in Figure 6-16.

Receiver equalization training is implementation specific.

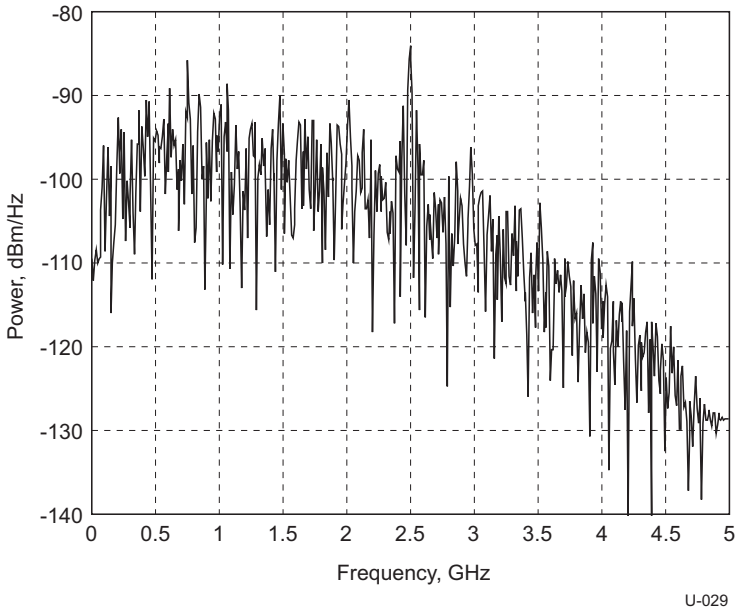


Figure 6-16. Frequency Spectrum of TSEQ

6.8.2 Informative Receiver CTLE Function

USB 3.0 allows the use of receiver equalization to meet system timing and voltage margins. For long cables and channels the eye at the Rx is closed, and there is no meaningful eye without first applying an equalization function. The Rx equalizer may be required to adapt to different channel losses using the Rx EQ training period. The exact Rx equalizer and training method is implementation specific.

The equation for the continuous time linear equalizer (CTLE) used to develop the specification is the compliance Rx EQ transfer function described below.

$$H(s) = \frac{A_{dc}\omega_{p1}\omega_{p2}}{\omega_z} \cdot \frac{s+\omega_z}{(s+\omega_{p1})(s+\omega_{p2})} \tag{10}$$

$$\begin{aligned} A_{dc} &= 0.667 \\ \omega_z &= 2\pi(650x10^6) \\ \omega_{p1} &= 2\pi(1.95x10^9) \\ \omega_{p2} &= 2\pi(5x10^9) \end{aligned} \tag{11}$$

Figure 6-17 is a plot of the Compliance EQ transfer function.

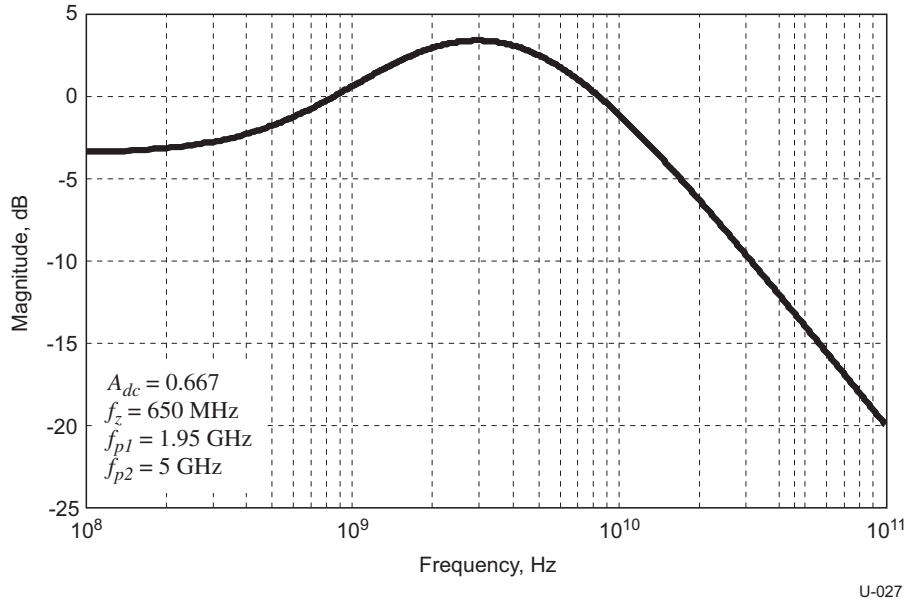


Figure 6-17. Tx Compliance Rx EQ Transfer Function

6.8.3 Receiver Electrical Parameters

Normative specifications are to be measured at the connector. Peak (p) and peak- peak (p-p) are defined in Section 6.6.2.

Table 6-13. Receiver Normative Electrical Parameters

Symbol	Parameter	5.0 GT/s	Units	Comments
UI	Unit Interval	199.94 (min) 200.06 (max)	ps	UI does not account for SSC caused variations.
R_{RX-DC}	Receiver DC common mode impedance	18 (min) 30 (max)	Ω	DC impedance limits are needed to guarantee Receiver detect. Measured with respect to ground over a voltage of 500 mV maximum.
$R_{RX-DIFF-DC}$	DC differential impedance	72 (min) 120 (max)	Ω	Measured with respect to ground over the maximum range of the differential signal (1200 mV).
$Z_{RX-HIGH-IMP-DC-POS}$[1]	DC Input CM Input Impedance for V>0 during Reset or power down	25 k (min)	Ω	Rx DC CM impedance with the Rx terminations not powered, measured over the range 0 – 500 mV with respect to ground.
$V_{RX-LFPS-DET-DIFFp-p}$	LFPS Detect Threshold	100 (min) 300 (max)	mV	Below the minimum is noise. Must wake up above the maximum.

Note

1. Only DC Input CM Input Impedance for V >0 is specified. DC Input CM Input Impedance for V <0 is not guaranteed and could be as low as 0 Ω.

The values in Table 6-14 are informative and not normative. They are included in this document to provide some guidance beyond the normative requirements in Table 6-13 for receiver design and development. A receiver can be fully compliant with the normative requirements of the specification and not meet all the values in this table (many of which are not measurable in a finished product). Similarly, a receiver that meets all the values in this table is not guaranteed to be in full compliance with the normative part of this specification.

Table 6-14. Receiver Informative Electrical Parameters

Symbol	Parameter	5.0 GT/s	Units	Comments
$V_{RX-DIFF-PP-POST-EQ}$	Differential Rx peak-to-peak voltage	30 (min)	mV	Measured after the Rx EQ function (Section 6.8.2)
t_{RX-TJ}	Max Rx inherent timing error	0.45 (max)	UI	Measured after the Rx EQ function (Section 6.8.2)
$t_{RX-DJ-DD}$	Max Rx inherent deterministic timing error	0.285 (max)	UI	Maximum Rx inherent deterministic timing error
$C_{RX-PARASITIC}$	Rx input capacitance for return loss	1.1 (max)	pf	

Symbol	Parameter	5.0 GT/s	Units	Comments
$V_{RX-CM-AC-P}$	Rx AC common mode voltage	150 (max)	mV Peak	Measured at Rx pins into a pair of 50 Ω terminations into ground. Includes Tx and channel conversion, AC range up to 5 GHz
$V_{RX-CM-DC-ACTIVE-IDLE-DELTA\_P}$	Rx AC common mode voltage during the U1 to U0 transition	200 (max)	mV Peak	Measured at Rx pins into a pair of 50 Ω terminations into ground. Includes Tx and channel conversion, AC range up to 5 GHz

6.8.4 Receiver Loopback

The entry and exit process for receiver loopback is described in Chapter 7.

Receiver loopback must be retimed. Direct connection from the Rx amplifier to the transmitter is not allowed for loopback mode. The receiver must continue to process SKPs as appropriate. SKP symbols must be consumed or inserted as required for proper clock tolerance compensation. Over runs or under runs of the clock tolerance buffers will reset the buffers to the neutral position.

During loopback the receiver must process the Bit Error Rate Test (BERT) commands.

Loopback must occur in the 10-bit domain. No error correction is allowed. All symbols must be transmitted as received with the exception of SKP and BERT commands.

6.8.4.1 Loopback BERT

During loopback the receiver processes the BERT ordered sets BRST, BDAT, and BERC. These ordered sets are described in Table 6-15 through Table 6-18. BRST and BDAT are looped back as received. BERC ordered sets are not looped back but are replaced with BCNT ordered sets. Any time a BRST is received, the error count register EC is set to 0 and the scrambling LFSR is set to 0FFFFh. Any number of consecutive BRST ordered sets may be received.

BRST followed by BDAT starts the bit error rate test. The BDAT sequence is the output of the scrambler and is equivalent to the logical idle sequence. It consists of scrambled 0 as described in Appendix B. As listed in Appendix B, the first 16 characters of the sequence are reprinted here:

FF	17	C0	14	B2	E7	02	82	72	6E	28	A6	BE	6D	BF	8D

The receiver must compare the received data to the BDAT sequence. Errors increment the error count register (EC) by 1. EC may not roll over but must be held at FFh. The LFSR is advanced once for every character except SKPs. The LFSR rolls over after 2^{16}-1 symbols. SKPs must be inserted or deleted as necessary for clock tolerance compensation.

The BERC command does not increment the error count register. The LFSR is advanced. The BERC ordered set is replaced by the BCNT ordered set. The BCNT ordered set includes the non-scrambled 8b/10b encoded error count (EC) register based on the running disparity. Following the return of the BCNT ordered set, the loopback slave shall continue to repeat symbols as received.

BERC may be sent multiple times. The EC register is not cleared by BERC ordered sets.

BERT continues until the loopback mode is terminated as described in Chapter 7.

Table 6-15. BRST

Symbol Number	Encoded Values	Description
0	K28.5	COM
1	K28.7	BRST

Table 6-16. BDAT

Symbol Number	Encoded Values	Description
D0.0 <0:n>	Logical Idle (refer to Appendix B)	Scrambled 0 Rolls over after 2^{16} -1 symbols

Table 6-17. BERC

Symbol Number	Encoded Values	Description
0	K28.3	BERC
1	K28.3	BERC
2	K28.3	BERC
3	K28.3	BERC

Table 6-18. BCNT

Symbol Number	Encoded Values	Description
0	K28.3	BERC
0	K28.3	BERC
EC<0:7>	DCODE	Error count (not scrambled)
EC<0:7>	DCODE	Error count (not scrambled)

The loopback BERT error count register and associated control mechanisms are optional. A device that does not implement the loopback BERT error count loops back BERC unmodified.

6.8.5 Normative Receiver Tolerance Compliance Test

The receiver tolerance test is tested in the compliance reference channel. A pattern generator will send a compliance test pattern with added jitter through the compliance reference channels to the receiver. The receiver will loop back the data and any difference in the pattern sent from the pattern generator and returned will be an error. When running the compliance tests, the receiver should be put into loopback mode.

Additional details on the receiver compliance test are contained in the reference document, *USB SuperSpeed Compliance Methodology*.

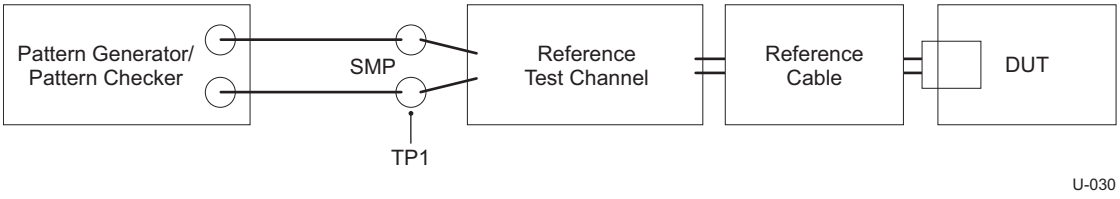


Figure 6-18. Rx Tolerance Setup

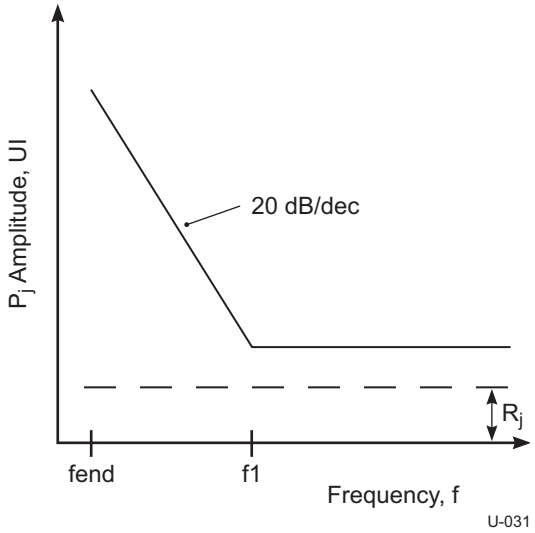


Figure 6-19. Jitter Tolerance Curve

The jitter components used to test the receiver shall meet the requirements of Table 6-19.

Table 6-19. Input Jitter Requirements for Rx Tolerance Testing

Symbol	Parameter	Value	Units	Notes
f1	Tolerance corner	4.9	MHz	
J_{Rj}	Random Jitter	0.0121	UI rms	1
$J_{Rj\_p\text{-}p}$	Random Jitter peak- peak at 10^{-12}	0.17	UI p-p	1,4
J_{Pj_500kHZ}	Sinusoidal Jitter	2	UI p-p	1,2,3
J_{Pj_1Mhz}	Sinusoidal Jitter	1	UI p-p	1,2,3
J_{Pj_2MHz}	Sinusoidal Jitter	0.5	UI p-p	1,2,3
J_{Pj_f1}	Sinusoidal Jitter	0.2	UI p-p	1,2,3
J_{Pj_50MHz}	Sinusoidal Jitter	0.2	UI p-p	1,2,3
V_full_swing	Transition bit differential voltage swing	0.75	V p-p	1
V_EQ_level	Non transition bit voltage (equalization)	-3	dB	1

Notes:

1. All parameters measured at TP1. The test point is shown in Figure 6-18.
2. Due to time limitations at compliance testing, only a subset of frequencies can be tested. However, the Rx is required to tolerate Pj at all frequencies between the compliance test points.
3. During the Rx tolerance test, SSC is generated by test equipment and present at all times. Each J_{Pj} source is then added and tested to the specification limit one at a time.
4. Random jitter is also present during the Rx tolerance test, though it is not shown in Figure 6-19.

6.9 Low Frequency Periodic Signaling (LFPS)

Low frequency periodic signaling (LFPS) is used for side band communication between the two ports across a link that is in a low power link state. It is also used when a link is under training, or when a downstream port issues Warm Reset to reset the link.

6.9.1 LFPS Signal Definition

Table 6-20 defines the LFPS electrical specification at the transmitter. An example differential LFPS waveform is shown in Figure 6-20. tPeriod is the period of an LFPS cycle. An LFPS burst is the transmission of continuous LFPS signal over a period of time defined by tBurst. An LFPS sequence is defined by the transmission of a single LFPS burst of duration tBurst over a period of time defined by tRepeat. The link is in electrical idle between the two contiguous LFPS bursts.

An LFPS message is encoded based on the variation of tBurst. tRepeat is defined as a time interval when the next LFPS message is transmitted. The LFPS messages include Polling.LFPS and Ping.LFPS, as defined in Table 6-21. There are also LFPS signaling defined for U1/U2 and Loopback exit, U3 wakeup, and Warm Reset.

The detailed use of LFPS signaling is specified in the following sections and Chapter 7.

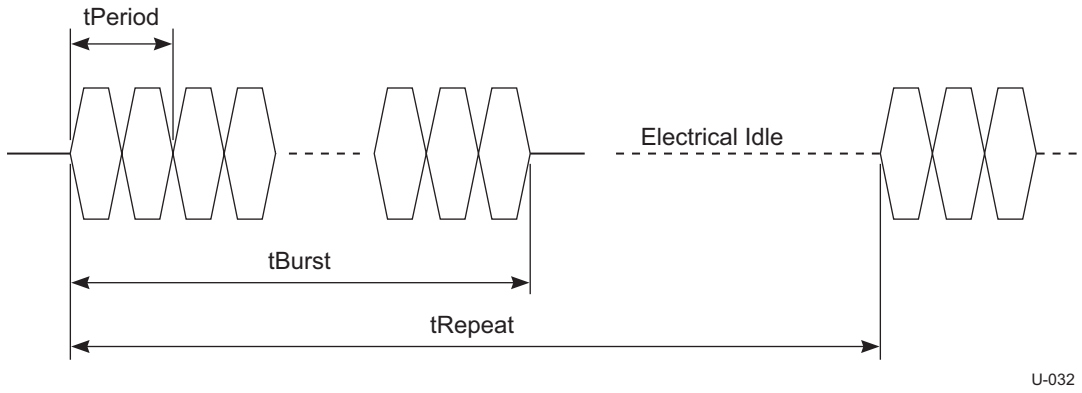


Figure 6-20. LFPS Signaling

Table 6-20. Normative LFPS Electrical Specification

Symbol	Minimum	Typical	Maximum	Units	Comments
tPeriod	20		100	ns	
$V_{CM-AC-LFPS}$			$V_{TX-CM-AC-PP-ACTIVE}$	mV	See Table 6-11
$V_{CM-LFPS-Active}$			10	mV	
$V_{TX-DIFF-PP-LFPS}$	800		1200	mV	Peak-peak differential amplitude
$V_{TX-DIFF-PP-LFPS-LP}$	400		600	mV	Low power peak-peak differential amplitude
tRiseFall2080			4	ns	Measured at compliance TP1, as shown in Figure 6-14.
Duty cycle	40		60	%	Measured at compliance TP1, as shown in Figure 6-14.

Table 6-21. LFPS Transmitter Timing[1]

	tBurst				tRepeat		
	Min	Typ	Max	Minimum Number of LFPS Cycles[2]	Min	Typ	Max
Polling.LFPS	0.6 µs	1.0 µs	1.4 µs		6 µs	10 µs	14 µs
Ping.LFPS[3]	40 ns		200 ns	2	160 ms	200 ms	240 ms
tReset[4,5]	80 ms	100 ms	120 ms				
U1 Exit[4,5]	600 ns[7]		2 ms				
U2/Loopback Exit[4,5]	80 µs[6]		2 ms				
U3 Wakeup[4,5]	80 µs[6]		10 ms				

Notes:

1. If the transmission of an LFPS signal does not meet the specification, the receiver behavior is undefined.
2. Only Ping.LFPS has a requirement for minimum number of LFPS cycles.
3. The declaration of Ping.LFPS depends on only the Ping.LFPS burst.
4. Warm Reset, U1/U2/Loopback Exit, and U3 Wakeup are all single burst LFPS signals. tRepeat is not applicable.
5. The minimum duration of an LFPS burst must be transmitted as specified. The LFPS handshake process and timing are defined in Section 6.9.2.
6. A Port in U2 or U3 is not required to keep its transmitter DC common mode voltage. When a port begins U2 exit or U3 wakeup, it may start sending LFPS signal while establishing its transmitter DC common mode voltage. To make sure its link partner receives a proper LFPS signal, a minimum of 80 µs tBurst must be transmitted. The same consideration also applies to a port receiving LFPS U2 exit or U3 wakeup signal.
7. A port is still required to detect U1 LFPS exit signal at a minimum of 300 ns. The extra 300 ns is provided as the guard band for successful U1 LFPS exit handshake.

IMPLEMENTATION NOTE

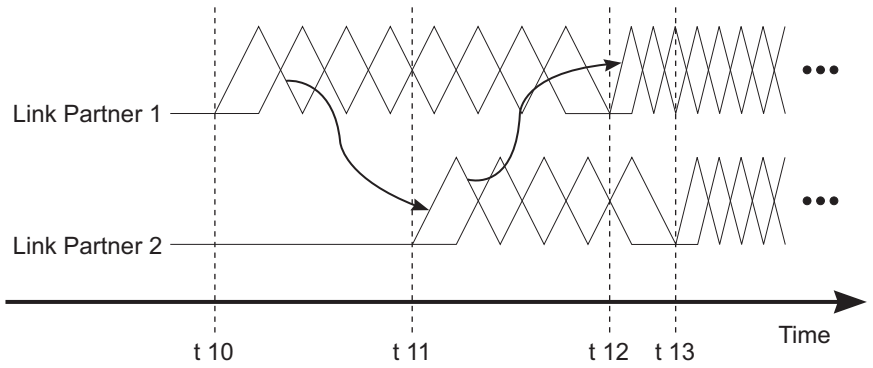
Detect and Differentiate Between Ping.LFPS and U1 LFPS Exit Signaling for a Downstream Port in U1 or U2

When a downstream port is in U1, it may receive either a Ping.LFPS as a message from its link partner to inform its presence, or an U1 LFPS exit signal to signal that its link partner is attempting exit from U1. This will also occur when a downstream port is in U2, since there are situations where a downstream port enters U2 from U1 when its U2 inactivity timer times out, and its link partner is still in U1.

Upon detecting the break of electrical idle due to receiving an LFPS signal, a downstream port may start a timer to measure the duration of the LFPS signal. If an electrical idle condition does not occur when the timer expires at 300 ns, a downstream port can declare the received LFPS signal is U1 exit and then respond to U1 exit by sending U1 LFPS exit handshake signal. If an electrical idle condition is detected before the timer reaches 300 ns, a downstream port can declare that the received LFPS signal is Ping.LFPS.

6.9.2 Example LFPS Handshake for U1/U2 Exit, Loopback Exit, and U3 Wakeup

The LFPS signal used for U1/U2 exit, Loopback exit, and U3 wakeup is defined the same as continuous LFPS signals with the exception of timeout values defined in Table 6-22. The handshake process for U1/U2 exit and U3 wakeup is illustrated in Figure 6-21. The timing requirements are different for U1 exit, U2 exit, Loopback exit, and U3 wakeup. They are listed in Table 6-22.



t10 – When Link Partner 1 starts to transmit LFPS, in initiating exit from either U1/U2/U3/Loopback.
t11 – When Link Partner 2 validates the received LFPS for the required t11-t10 duration and starts to transmit LFPS in response.
t12 – When Link Partner 1 validates the received LFPS of the required t12-t11 and t12-t10 durations and starts to transmit SS signaling.
t13 – When Link Partner 2 transmits LFPS of meeting the required t13-t11 duration and starts to transmit SS signaling.

U-033A

Figure 6-21. U1 Exit, U2 Exit, and U3 Wakeup LFPS Handshake Timing Diagram

Note: the timing diagram in Figure 6-21 is for illustration of the LFPS handshake process only.

The handshake process is as follows:

- Link partner 1 initiates exit by transmitting LFPS at time t10 (see Figure 6-21). LFPS transmission shall continue until the handshake is declared either successful or failed.
- Link partner 2 detects valid LFPS on its receiver and responds by transmitting LFPS at time t11. LFPS transmission shall continue until the handshake is declared either successful or failed.
- A successful handshake is declared for link partner 1 if the following conditions are met within "tNoLFPSResponseTimeout" after t10 (see Figure 6-21 and Table 6-22):
  1. Valid LFPS is received from link partner 2.

     Note: In case of concurrent U1 exit, where both ports initiate U1 exit simultaneously, both ports will assume to be Link partner 1. Both ports will start receiving LFPS signal before t10. And received U1 LFPS exit signal may be validated around t10. This may result in a minimum duration of U1 exit LFPS signal. To ensure a successful U1 exit under such situations, both ports shall transmit U1 LFPS exit signal for 600 ns before exiting U1.

2. For U1 exit, U2 exit, U3 Wakeup and not Loopback exit, link partner 1 is ready to transmit the training sequences and the maximum time gap after an LFPS transmitter stops transmission and before a SuperSpeed transmitter starts transmission is 20 ns.

Note: There is no Near End Cross Talk (NEXT) specification for SuperSpeed transmitters and receivers. Therefore, when a port enters Recovery and starts transmitting TS1 Ordered Sets and its link partner is in electrical idle after successful LFPS handshake, a port may potentially train its receiver using its own TS1 Ordered Sets due to NEXT. The intention of adding the second exit condition is to prevent a port from electrical idle before transitioning to Recovery.

- A successful handshake is declared for link partner 2 if the following conditions are met:
  1. Link partner 2 has transmitted the minimum LFPS defined as (t13 – t11) in Table 6-22.
  2. For U1 exit, U2 exit, U3 Wakeup, and not Loopback exit, link partner 2 is ready to transmit the training sequences and the maximum time gap after an LFPS transmitter stops transmission and before a SuperSpeed transmitter starts transmission is 20 ns.
- A U1 exit, U2 exit, Loopback exit, and U3 wakeup handshake failure shall be declared if the conditions for a successful handshake are not met.
- Link partner 1 shall declare a failed handshake if its successful handshake conditions were not met.
- Link partner 2 shall declare a failed handshake if its successful handshake conditions were not met.

Note: Except for Ping.LFPS, when an upstream port in Ux or Loopback.Active receives an LFPS signal, it shall proceed with U1/U2 exit, or U3 wakeup, or Loopback exit handshake even if the LFPS is later determined to be a Warm Reset. If the LFPS is a Warm Reset, an upstream port, if in Ux, will enter Recovery and then times out to SS.Inactive, or if in Loopback Active, will enter Rx.Detect and then transitions to Polling.LFPS. When Warm Reset is detected, an upstream port will enter Rx.Detect.

Table 6-22. LFPS Handshake Timing for U1/U2 Exit, Loopback Exit, and U3 Wakeup

	U1 Exit		U2/Loopback Exit		U3 Wakeup	
	Min	**Max**	**Min**	**Max**	**Min**	**Max**
t11 – t10	0.3 μs	0.9 μs/2 ms[1] – 900 ns	0.3 μs	2 ms	0.3 μs	10 ms
t13 – t10	0.9 μs	2 ms		2 ms		20 ms
t12 – t11	0.3 μs	0.9 μs[1]	0	2 ms	0	10 ms
t13 – t11	0.6 μs	0.9 μs[1]	80 μs	2 ms	80 μs	10 ms
t12 – t10	0.6 μs	2 ms[1]	80 μs	2 ms	80 μs	10 ms
tNoLFPSResponseTimeout		2 ms		2 ms		10 ms

Note:

1. There are two sets of maximum timing requirements. The set with short timing requirement applies to normal operations when U2_Inactivity_Timer is disabled. The set with relaxed timing requirement applies to operations when U2_Inactivity_Timer is enabled. It also includes one corner case where U2_Inactivity_Timer is disabled and the port, upon entry to U1, initiates U1 exit immediately.
2. In a case where U2_Inactivity_Timer is enabled, it is the responsibility of each link partner to respond accordingly depending on its U1 or U2 state. For example, when link partner 1 initiates exit in U1 and link partner 2 is in U2, it is expected that both link partners will eventually enter U0 with respective timings starting from different U1/U2 states. Essentially, t12-t10 of link partner 1 follows U1 Exit tBurst timing and t13-t11 of link partner 2 follows U2 Exit tBurst timing.

6.9.3 Warm Reset

A Warm Reset is a reset generated only by a downstream port to an upstream port. A downstream port may issue a Warm Reset at any Link states except SS.Disabled. An upstream port is required to detect a Warm Reset at any link states except SS.Disabled.

Note: Warm Reset is defined to be able to reset a hardware failure of a device, such as the LTSSM hanging. Under this assumption, Warm Reset may be detected in any link states except SS.Disabled.

A Warm Reset shares the same continuous LFPS signal as a low power Link state exit handshake signal. In order for an upstream port to be able to differentiate between the two signals, the tBurst of a Warm Reset is extended, as is defined in Table 6-20.

The Warm Reset assertion is asynchronous between a downstream port and an upstream port since it has to take a certain period of time for an upstream port to declare that a Warm Reset is detected. However, the de-assertion of the Warm Reset between a downstream port and an upstream port must be made synchronous. Figure 6-22 shows a timing diagram of Warm Reset generation and detection when a port is U3. Once a Warm Reset is issued by a downstream port, it will take at least tResetDelay for an upstream port to declare the detection of Warm Reset. Once a Warm Reset is detected, an upstream port must continue to assert the Warm Reset until it no longer receives any LFPS signals from a downstream port.

- An upstream port shall declare the detection of Warm Reset within tResetDelay. The minimum tResetDelay shall be 18 ms; the maximum tResetDelay shall be 50 ms.

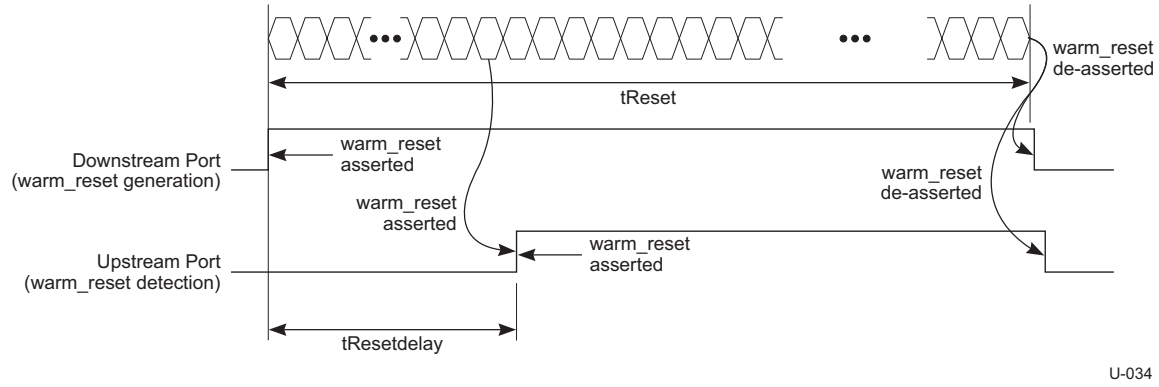


Figure 6-22. Example of Warm Reset Out of U3

6.10 Transmitter and Receiver DC Specifications

6.10.1 Informative ESD Protection

All signal and power pins must withstand 2000 V of ESD using the human body model (CLASS 2 per JEDEC JESD22-A114F) and 500 V using the charged device model (CLASS III per JEDEC JESD22-C101D) without damage.

6.10.2 Informative Short Circuit Requirements

All Transmitters and Receivers must support surprise hot insertion/removal without damage to the component. The Transmitter and Receiver must be capable of withstanding sustained short circuit to ground of Txp (Rxp) and Txn (Rxn).

6.10.3 Normative High Impedance Reflections

During an asynchronous reset event, one device may be reset while the other device is transmitting. The device under reset is required to disconnect the receiver termination. During this time, the device under reset may be receiving active data. Since the data is not terminated, the differential voltage into the receiver will be doubled. For a short channel, the receiver may experience a total of 2* $V_{DIFF.}$

The receiver must tolerate this doubling of the negative voltage that can occur if the Rx termination is disconnected. A part must tolerate a 20 ms event that doubles the voltage on the receiver input when the termination is disconnected 10,000 times over the life time of the part.

6.11 Receiver Detection

6.11.1 Rx Detect Overview

The Receiver Detection circuit is implemented as part of a Transmitter and must correctly detect whether a load impedance equivalent to a DC impedance $R_{RX\text{-}DC}$ (Table 6-13) is present. The Rx detection operates on the principle of the RC time constant of the circuit. This time constant changes based on the presence of the receiver termination. This is conceptually illustrated in Figure 6-23. In this figure, R_Detect is the implementation specific charging resistor. C_AC is the AC capacitor that is in the circuit only if R_Term is also present, otherwise, only C_Parasitic is present.

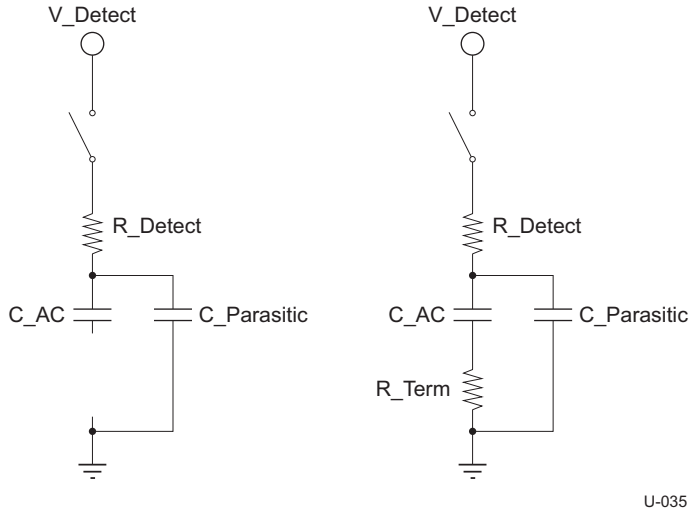


Figure 6-23. Rx Detect Schematic

The left side of Figure 6-23 shows the Receiver Detection circuit with no termination present. The right side of the figure is the same circuit with termination.

Detect voltage transition must be common mode. Detect voltage transition must conform to $V_{TX\_RCV\_DETECT}$ as described in Table 6-10.

The receiver detect sequence must be in the positive common mode direction only. Negative receiver detection is not allowed.

6.11.2 Rx Detect Sequence

The recommended behavior of the Receiver Detection sequence is:

1. A Transmitter must start at a stable voltage prior to the detect common mode shift.
2. A Transmitter changes the common mode voltage on Txp and Txn consistent with detection of Receiver high impedance which is bounded by parameter $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}POS}$ listed in Table 6-13.
3. A Receiver is detected based on the rate that the lines change to the new voltage.
   - The Receiver is not present if the voltage at the Transmitter charges at a rate dictated only by the Transmitter impedance and the capacitance of the interconnect and series capacitor.
   - The Receiver is present if the voltage at the Transmitter charges at a rate dictated by the Transmitter impedance, the series capacitor, the interconnect capacitance, and the Receiver termination.

Any time Electrical Idle is exited the detect sequence does not have to execute or may be aborted. During the Device connect, the Device receiver has to guarantee it is always in high impedance state while its power plane is stabilizing. This is required to avoid the Host falsely detecting the Device and starting the training sequence before the Device is ready. Similarly a disabled port has to keep its receiver termination in high impedance which is bounded by parameters $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}POS}$ until directed by higher layer to exit from the Disabled state. In contrast, a port which is at U1/U2/U3 Electrical Idle must have its Receiver Termination turned on and meet the $R_{RX\text{-}DC}$ specification.

6.11.3 Upper Limit on Channel Capacitance

The interconnect total capacitance to ground seen by the Receiver Detection circuit must not exceed 3 nF to ground, including capacitance added by attached test instrumentation. This limit is needed to guarantee proper operation during Receiver detect. Note that this capacitance is separate and distinct from the AC coupling capacitance value.

7 Link Layer

The link layer has the responsibility of maintaining the link connectivity so that successful data transfers between the two link partners are ensured. A robust link flow control is defined based on packets and link commands. Packets are prepared in the link layer to carry data and different information between the host and a device. Link commands are defined for communications between the two link partners. Packet frame ordered sets and link command ordered sets are also constructed such that they are tolerant to one symbol error. In addition, error detections are also incorporated into a packet and a link command to verify packet and link command integrity.

The link layer also facilitates link training, link testing/debugging, and link power management. This is accomplished by the introduction of Link Training Status State Machine (LTSSM).

The focus of this chapter is to address the following in detail:

- Packet Framing
- Link command definition and usage
- Link initialization and flow control
- Link power management
- Link error rules/recovery
- Resets
- LTSSM specifications

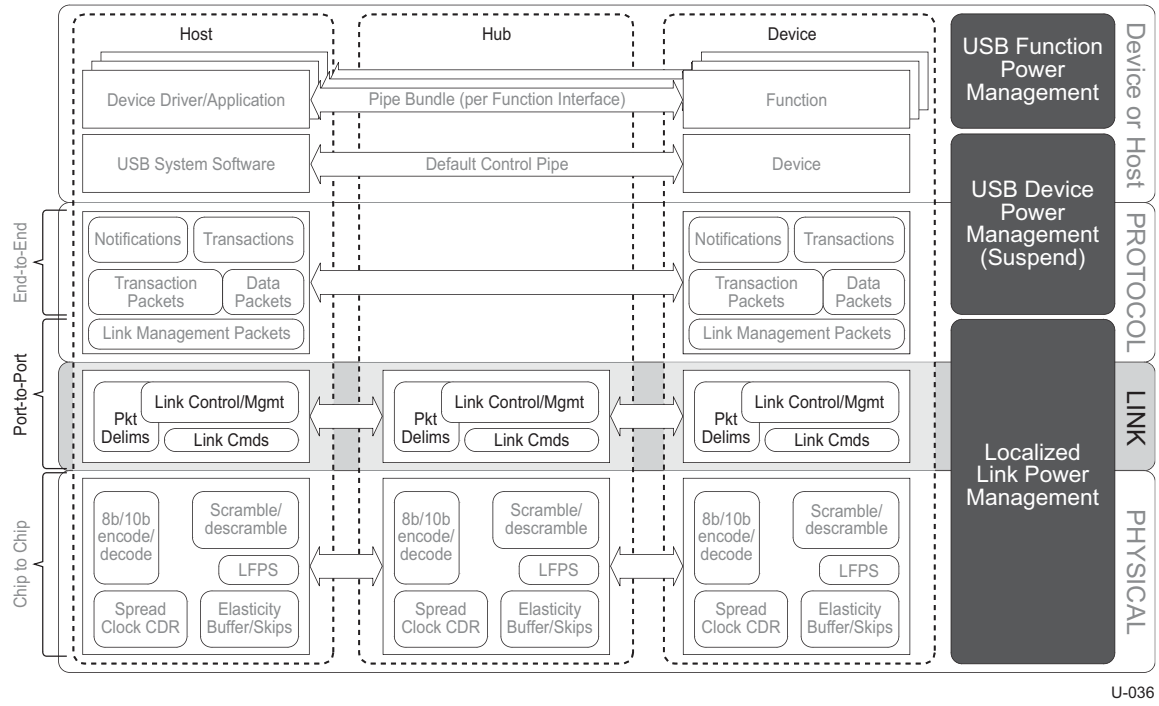


Figure 7-1. Link Layer

7.1 Byte Ordering

Multiple byte fields in a packet or a link command are moved over to the bus in little-endian order, i.e., the least significant byte (LSB) first, and the most significant byte (MSB) last. Figure 7-2 shows an example of byte ordering.

Each byte of a packet or link command will be encoded in the physical layer using 8b/10b encoding. Refer to Section 6.3 regarding 8b/10b encoding and bit ordering.

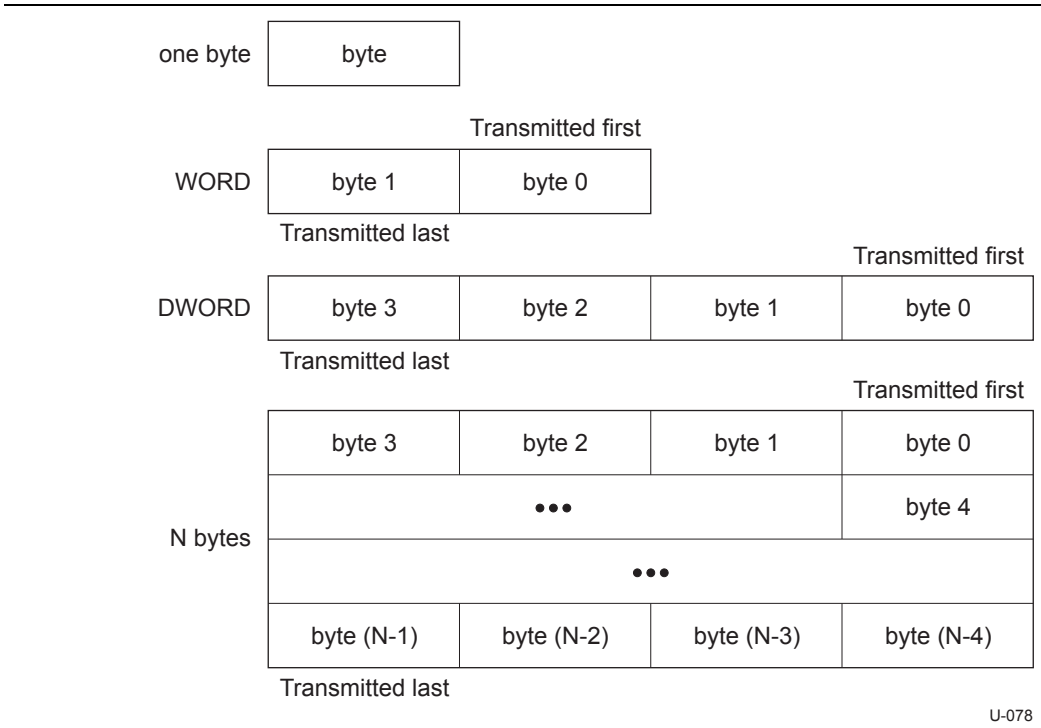


Figure 7-2. SuperSpeed Byte Ordering

7.2 Link Management and Flow Control

This section contains information regarding link data integrity, flow control, and link power management.

- The packet and packet framing section defines packet types, packet structures, and CRC requirements for each packet.
- The link command section defines special link command structures that control various functionalities at the link level.
- The logical idle defines a special symbol used in U0.
- The flow control defines a set of handshake rules for packet transactions.

7.2.1 Packets and Packet Framing

SuperSpeed uses packets to transfer information. Detailed packet formats for Link Management Packets (LMP), Transaction Packets (TP), Isochronous Timestamp Packets (ITP), and Data Packets (DP) are defined in Section 8.2.

7.2.1.1 Header Packet Structure

All header packets are 20 symbols long, as is formatted in Figure 7-3. This includes LMPs, TPs, ITPs, and DPHs. A header packet consists of three parts, a header packet framing, a packet header, and a Link Control Word.

7.2.1.1.1 Header Packet Framing

Header packet framing, HPSTART ordered set, is a four-symbol header packet starting frame ordered set based on K-symbols. It is defined as three consecutive symbols of SHP followed by a K-symbol of EPF. A header packet shall always begin with HPSTART ordered set. The construction of the header packet framing is to achieve one symbol error tolerance.

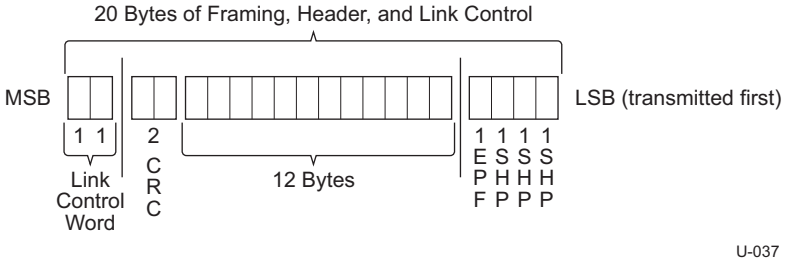


Figure 7-3. Header Packet with HPSTART, Packet Header, and Link Control Word

7.2.1.1.2 Packet Header

A packet header consists of 14 bytes as formatted in Figure 7-4. It includes 12 bytes of header information and a 2-byte CRC-16. CRC-16 is used to protect the data integrity of the 12-byte header information.

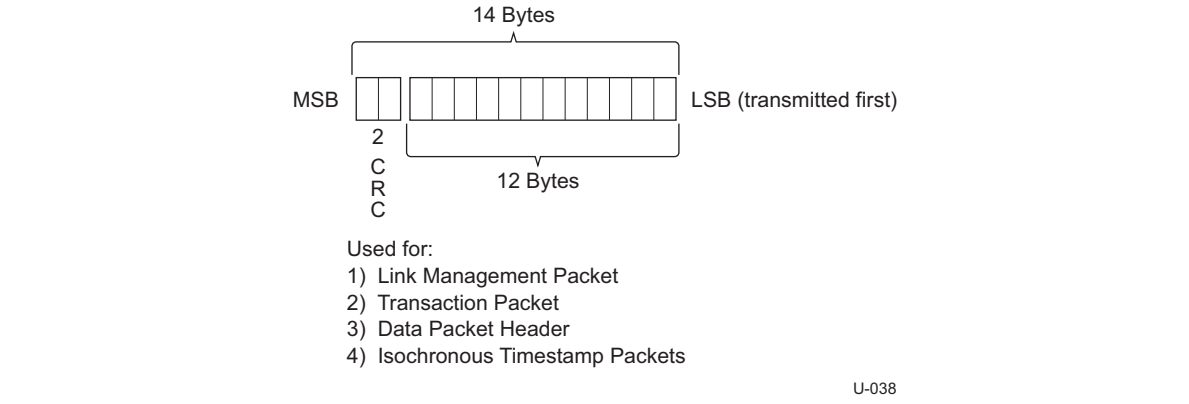


Figure 7-4. Packet Header

The implementation of CRC-16 on the packet header is defined below:

- The polynomial for CRC-16 shall be 100Bh.

  Note: The CRC-16 polynomial is not the same as the one used for USB 2.0.
- The initial value of CRC-16 shall be FFFFh.
- CRC-16 shall be calculated for all 12 bytes of the header information, not inclusive of any packet framing symbols.
- CRC-16 calculation shall begin at byte 0, bit 0 and continue to bit 7 of each of the 12 bytes.
- The remainder of CRC-16 shall be complemented.
- The residual of CRC-16 shall be F6AAh.

Note: The inversion of the CRC-16 remainder adds an offset of FFFFh that will create a constant CRC-16 residual of F6AAh at the receiver side.

Figure 7-5 is an illustration of CRC-16 remainder generation. The output bit ordering is listed in Table 7-1.

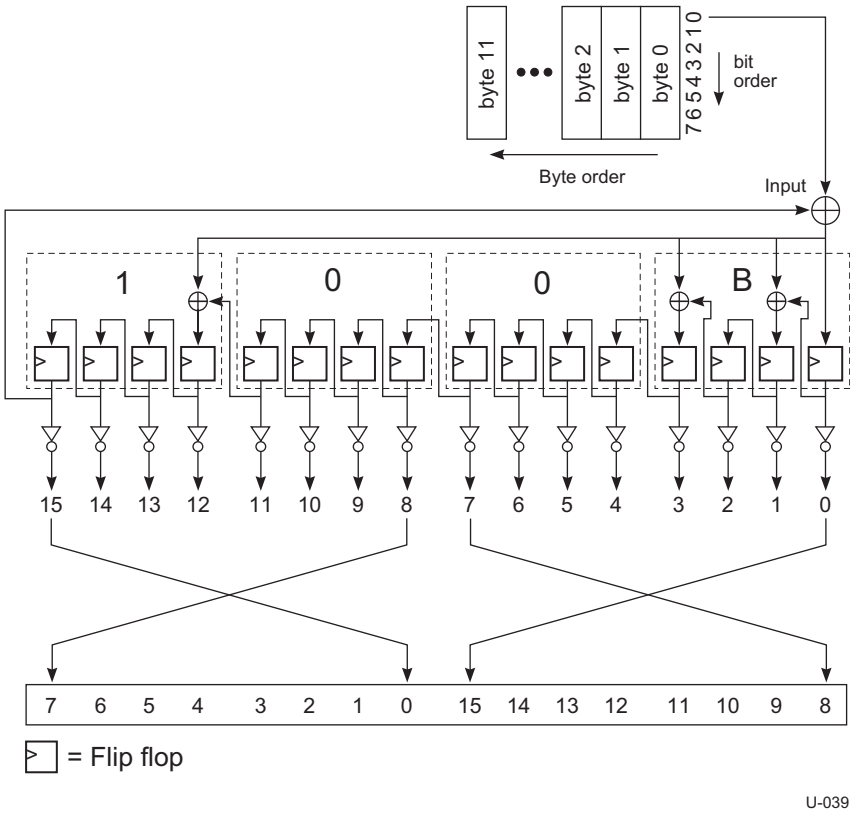


Figure 7-5. CRC-16 Remainder Generation

Table 7-1. CRC-16 Mapping

CRC-16 Result Bit	Position in CRC-16 Field
0	15
1	14
2	13
3	12
4	11
5	10
6	9
7	8
8	7
9	6
10	5
11	4
12	3
13	2
14	1
15	0

7.2.1.1.3 Link Control Word

The 2-byte Link Control Word is formatted as shown in Figure 7-6. It is used for both link level and end-to-end flow control.

The Link Control Word shall contain a 3-bit Header Sequence Number, 3-bit Reserved, a 3-bit Hub Depth Index, a Delayed bit (DL), a Deferred bit (DF), and a 5-bit CRC-5.

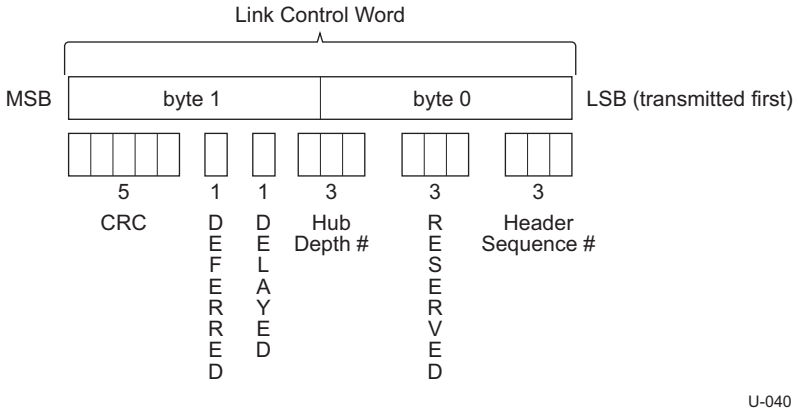


Figure 7-6. Link Control Word

CRC-5 protects the data integrity of the Link Control Word. The implementation of CRC-5 is defined below:

- The CRC-5 polynomial shall be 00101b.
- The Initial value for the CRC-5 shall be 11111b.
- CRC-5 is calculated for the remaining 11 bits of the Link Control Word.
- CRC-5 calculation shall begin at bit 0 and proceed to bit 10.
- The remainder of CRC-5 shall be complemented, with the MSb mapped to bit 11, the next MSb mapped to bit 12, and so on, until the LSb mapped to bit 15 of the Link Control Word.
- The residual of CRC-5 shall be 01100b.

  Note: The inversion of the CRC-5 remainder adds an offset of 11111b that will create a constant CRC-5 residual of 01100b at the receiver side.

Figure 7-7 is an illustration of CRC-5 remainder generation.

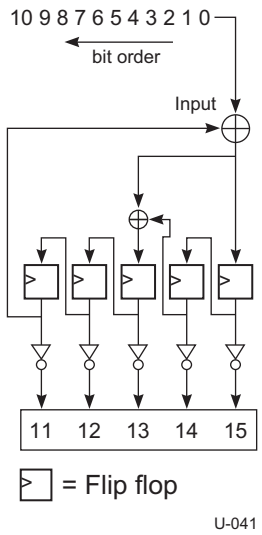


Figure 7-7. CRC-5 Remainder Generation

7.2.1.2 Data Packet Payload Structure

Data packets are a special type of packet consisting of a Data Packet Header (DPH) and a Data Packet Payload (DPP). The DPH is defined in Section 7.2.1.1. The DPP, on the other hand, consists of a data packet payload framing, and a variable length of data followed by 4 bytes of CRC-32. Figure 7-8 describes the format of a DPP.

7.2.1.2.1 Data Packet Payload Framing

DPP framing consists of eight K-symbols, a four-symbol DPP starting frame ordered set and a four-symbol DPP ending frame ordered set. As indicated by Figure 7-8, a DPPSTART ordered set, which is a DPP starting frame ordered set, consists of three consecutive K-symbols of SDP followed by a single K-symbol of EPF. A DPP ending frame ordered set has two different types. The first type, DPPEND ordered set, is a DPP ending frame ordered set which consists of three consecutive K-symbol of END followed by a single K-symbol of EPF. The second type, DPPABORT ordered set, is a DPP aborting frame ordered set which consists of three consecutive K-symbol of EDB (end of nullified packet) followed by a single K-symbol of EPF. The DPPEND ordered set is used to indicate a normal ending of a complete DPP. The DPPABORT ordered set is used to indicate an abnormal ending of a DPP.

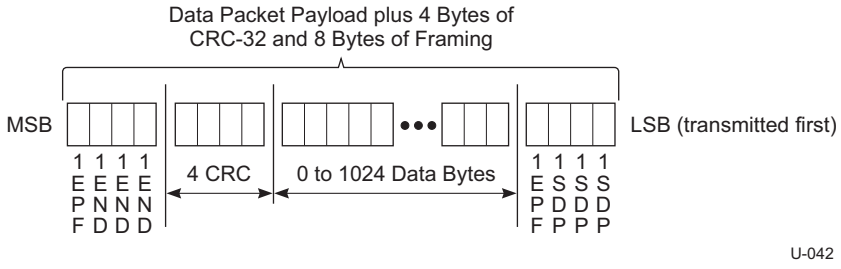


Figure 7-8. Data Packet Payload with CRC-32 and Framing

7.2.1.2.2 Data Packet Payload

The DPP section consists of 0 to 1024 data bytes followed by 4 bytes CRC-32. Any premature termination of a DPP shall contain a DPPABORT ordered set. A DPP shall immediately follow its corresponding DPH with no spacing in between.

CRC-32 protects the data integrity of the data payload. CRC-32 is as follows:

- The CRC-32 polynomial shall be 04C1 1DB7h.
- The CRC-32 Initial value shall be FFFF FFFFh.
- CRC-32 shall be calculated for all bytes of the DPP, not inclusive of any packet framing symbols.
- CRC-32 calculation shall begin at byte 0, bit 0 and continue to bit 7 of each of the bytes of the DPP.
- The remainder of CRC-32 shall be complemented.
- The residual of CRC-32 shall be C704DD7Bh.

Note: The inversion of the CRC-32 remainder adds an offset of FFFF FFFFh that will create a constant CRC-32 residual of C704DD7Bh at the receiver side.

Figure 7-9 is an illustration of CRC-32 remainder generation. The output bit ordering is listed in Table 7-2.

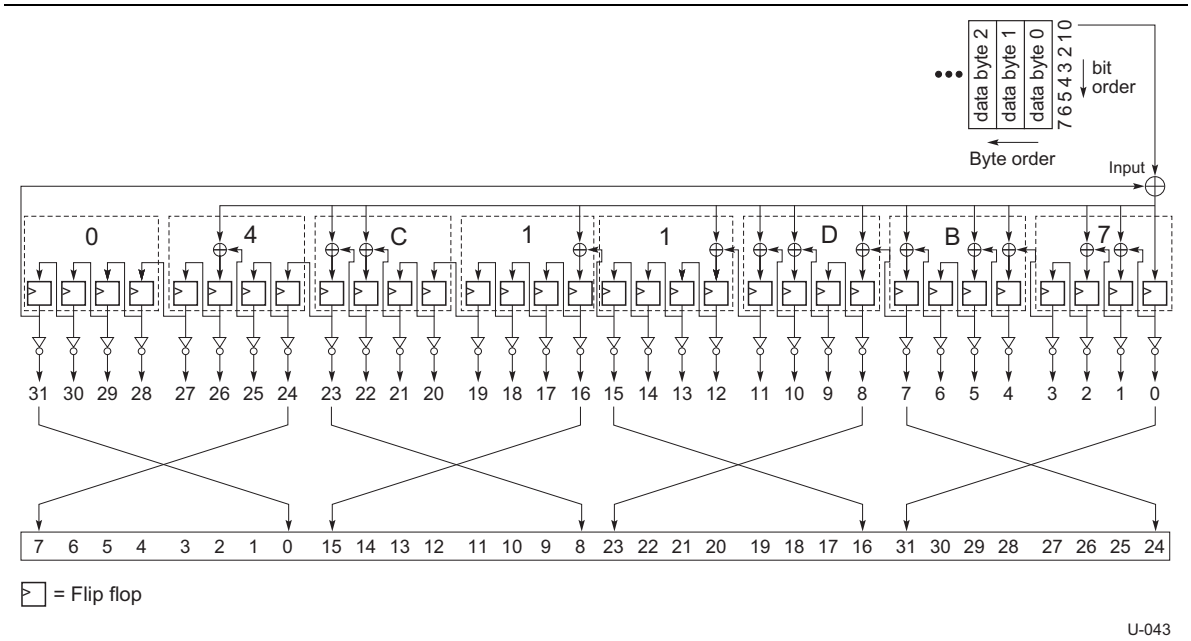


Figure 7-9. CRC-32 Remainder Generation

Table 7-2. CRC-32 Mapping

CRC-32 Result Bit	Position in CRC-32 Field
0	31
1	30
2	29
3	28
4	27
5	26
6	25
7	24
8	23
9	22
10	21
11	20
12	19
13	18
14	17
15	16
16	15
17	14
18	13
19	12
20	11
21	10
22	9
23	8
24	7
25	6
26	5
27	4
28	3
29	2
30	1
31	0

7.2.1.2.3 Spacing Between Data Packet Header and Data Packet Payload

There shall be no spacing between a DPH and its corresponding DPP. This is illustrated in Figure 7-10.

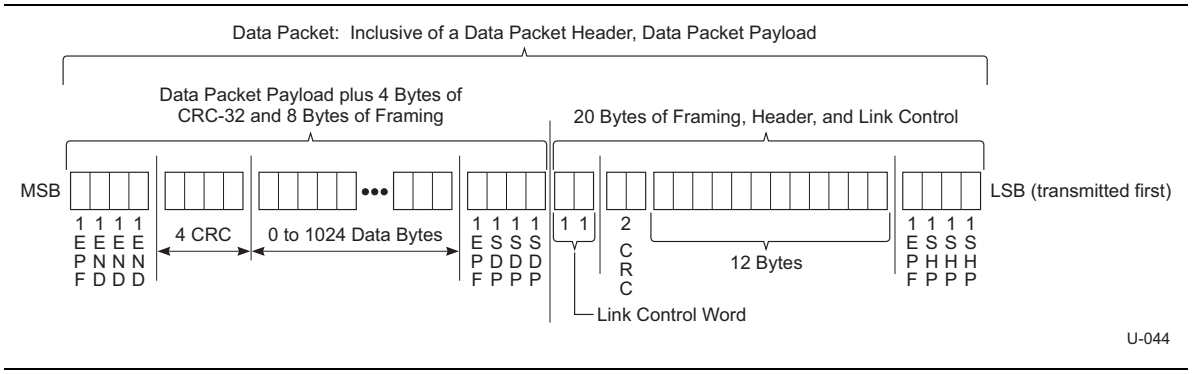


Figure 7-10. Data Packet with Data Packet Header Followed by Data Packet Payload

Additional details on how header packets are transmitted and received at the link level are described in Section 7.2.4.

7.2.2 Link Commands

Link commands are used for link level data integrity, flow control and link power management. Link commands are a fixed length of eight symbols and contain repeated symbols to increase the error tolerance. Refer to Section 7.3 for more details. Link command names have the L-preface to differentiate their link level usage and to avoid confusion with packets.

7.2.2.1 Link Command Structure

Link command shall be eight symbols long and constructed with the following format shown in Figure 7-11. The first four symbols, LCSTART, are the link command starting frame ordered set consisting of three consecutive SLCs followed by EPF. The second four symbols consist of a two-symbol link command word and its replica. Both link command words are scrambled. Table 7-3 summarizes the link command structure.

Table 7-3. Link Command Ordered Set Structure

Symbol Number	Description
0	SLC (Start Link Command)
1	SLC (Start Link Command)
2	SLC (Start Link Command)
3	EPF
4~5	Link Command Word
6~7	Link Command Word

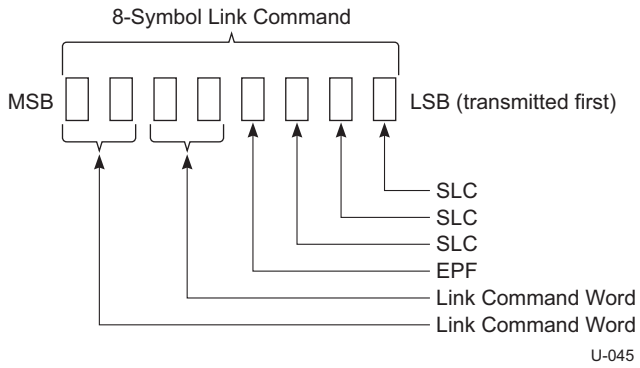


Figure 7-11. Link Command Structure

7.2.2.2 Link Command Word Definition

Link command word is 16 bits long with the 11-bit link command information protected by a 5-bit CRC-5 (see Figure 7-12). The 11-bit link command information is defined in Table 7-4. The calculation of CRC-5 is the same as Link Control Word illustrated in Figure 7-6.

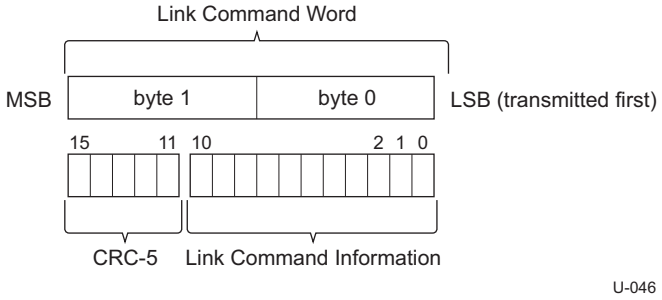


Figure 7-12. Link Command Word Structure

Table 7-4. Link Command Bit Definitions

Class		Type	b6~4	Sub-Type
b10~9	**Link Command**	**b8~7**		**b3~0**
00	LGOOD_n LRTY LBAD LCRD_x	00: LGOOD_n	Reserved (000)	b3: Reserved b2~0: HP Sequence Number 000: LGOOD_0 001: LGOOD_1 010: LGOOD_2 011: LGOOD_3 100: LGOOD_4 101: LGOOD_5 110: LGOOD_6 111: LGOOD_7
		01: LCRD_x		b3~2: Reserved b1~0: Rx Header Buffer Credit 00: LCRD_A 01: LCRD_B 10: LCRD_C 11: LCRD_D
		10: LRTY 11: LBAD		Reserved (0000)
01	LGO_Ux LAU LXU LPMA	00: LGO_Ux		0001: LGO_U1 0010: LGO_U2 0011: LGO_U3 Others: Reserved
		01: LAU 10: LXU 11: LPMA		Reserved (0000)
10	LDN LUP	00: LUP 11: LDN Others: Reserved		Reserved (0000)
11: Reserved	Reserved	Reserved (0000)		Reserved (0000)

Link commands are defined for four usage cases. First, link commands are used to ensure the successful transfer of a packet. Second, link commands are used for link flow control. Third, link commands are used for link power management. And finally, special link commands are defined for a port to signal its presence in U0.

Successful header packet transactions between the two link partners require proper header packet acknowledgement. Rx Header Buffer Credit exchange facilitates link flow control. Header packet acknowledgement and Rx Header Buffer Credit exchange are realized using different link commands. LGOOD_n (n = 0 to 7) and LBAD are used to acknowledge whether a header packet has been received properly or not. LRTY is used to signal that a header packet is re-sent. LCRD_A, LCRD_B, LCRD_C, and LCRD_D are used to signal the availability of Rx Header

Buffers in terms of Credit. In the following sections, LCRD_x is used with x denoting either A, B, C, or D. See Table 7-5 for details.

LGOOD_n uses an explicit numerical index called Header Sequence Number to represent the sequencing of a header packet. The Header Sequence Number starts from 0 and is incremented by one based on modulo-8 addition with each header packet. The index corresponds to the received Header Sequence Number and is used for flow control and detection of lost or corrupted header packets.

LCRD_x uses an explicit alphabetical index. The index A, B, C, D, A, B, C… is advanced by one with each header packet being processed and an Rx Header Buffer Credit is available. The index is used to ensure Rx Header Buffer Credits are received in order such that missing of an LCRD_x can be detected.

LBAD and LRTY do not use indexes.

LGO_U1, LGO_U2, LGO_U3, LAU, LXU, and LPMA are link commands used for link power management.

LDN and LUP are special link commands used by a downstream port and an upstream port to indicate their port presence in U0. The usage of LDN and LUP is described in Table 7-5.

Additional requirements and examples on the use of link commands are found in Section 7.2.4.

Table 7-5. Link Command Definitions

Link Command	Definition – See Sections 7.2.4.1, 7.2.4.2, and 7.5.6 for detailed use and requirements.
LGOOD_n	n (0, 1, 2, ...7): Header Sequence Number. Sent by a port receiving a header packet when all of the following conditions are true: • The header packet has a valid structure and can be recognized by the receiver. • CRC-5 and CRC-16 are valid. • Header Sequence Number in the received header packet matches the expected Rx Header Sequence Number. • An Rx Header Buffer in the receiver is available for storing the received header packet. Mismatch between a Header Sequence Number in the received header packet and the expected Rx Header Sequence Number will result in a port transitioning to Recovery. Received by a port sending a header packet. This is an acknowledgement from a link partner that a header packet with the Header Sequence Number of "n" is received properly. Receipt of LGOOD_n mismatching the expected ACK Tx Header Sequence Number will result in a port transitioning to Recovery. Also sent by a port upon entry to U0 as the Header Sequence Number Advertisement to initialize the ACK Tx Header Sequence Number of the two ports. Refer to Section 7.2.4.1 for details.
LBAD	Bad header packet. Sent by a port receiving the header packet in response to an invalid header packet. Packet that was received has corrupted CRC-5 and/or CRC-16. Receipt of LBAD will cause a port to resend all header packets after the last header packet that has been acknowledged with LGOOD_n. Refer to Section 7.2.4.1 for details.

Link Command	Definition – See Sections 7.2.4.1, 7.2.4.2, and 7.5.6 for detailed use and requirements.
LCRD_x	x (A, B, C, D): Rx Header Buffer Credit Index. Signifies that a single Rx Header Buffer Credit has been made available. Sent by a port after receiving a header packet that meets the following criteria: • LGOOD_n has been or will be sent. • The header packet has been processed, and an Rx Header Buffer Credit is available. LCRD_x is sent in the alphabetical order of A, B, C, D, and back to A without skipping. Missing LCRD_x will cause the link to transition to Recovery. Refer to Section 7.2.4.1 for details.
LRTY	Sent by a port before resending the first header packet in response to receipt of LBAD.
LGO_U1	Sent by a port requesting entry to U1.
LGO_U2	Sent by a port requesting entry to U2.
LGO_U3	Sent by a downstream port requesting entry to U3. An upstream port shall accept the request.
LAU	Sent by a port accepting the request to enter U1, U2, or U3.
LXU	Sent by a port rejecting the request to enter U1 or U2.
LPMA	Sent by a port upon receiving LAU. Used in conjunction with LGO_Ux and LAU handshake to guarantee both ports are in the same state.
LDN	Downstream port present in U0. Sent by a downstream port every 10 µs when there are no packets or other link commands to be transmitted. Refer to Section 7.5.6.1 for details.
LUP	Device present in U0. Sent by an upstream port every 10 µs when there are no packets or other link commands to be transmitted. Refer to Section 7.5.6.1 for details.

7.2.2.3 Link Command Placement

The link command placement shall meet the following rules:

- Link commands shall not be placed inside header packet structures (i.e., within LMPs, TPs, ITPs, or DPHs).
- Link commands shall not be placed within the DPP of a DP structure.
- Link commands shall not be placed between the DPH and the DPP.
- Link commands may be placed before and after a header packet with the exception that they shall not be placed in between a DPH and its DPP.
- Multiple link commands are allowed to be transmitted back to back.
- Link commands shall not be sent until all scheduled SKP ordered sets have been transmitted.

  Note: Additional rules regarding scheduling of link commands are found in Section 10.7.5 to Section 10.7.12.

7.2.3 Logical Idle

Logical Idle is defined to be a period of one or more symbol periods when no information (packets or link commands) is being transferred on the link. A special D-Symbol (00h), defined as Idle Symbol (IS), shall be transmitted by a port at any time in U0 meeting the logical idle definition.

The IS shall be scrambled according to rules described in Section 6.4.3.

Table 7-6. Logical Idle Definition

Symbol	Data Byte Name	Data Byte Value	Definition
IS	D0.0	00h	Represents Idle state on the bus.

7.2.4 Link Command Usage for Flow Control, Error Recovery, and Power Management

Link commands are used for link level header packet flow control, to identify lost/corrupted header packets and to initiate/acknowledge link level power management transitions. The construction and descriptions for each link command are found in Section 7.2.2.

7.2.4.1 Header Packet Flow Control and Error Recovery

Header packet flow control is used for all header packets. It requires each side of the link to follow specific header buffer and transmission ordering constraints to guarantee a successful packet transfer and link interoperability. This section describes, in detail, the rules of packet flow control.

7.2.4.1.1 Initialization

The link initialization refers to initialization of a port once a link transitions to U0 from Polling, Recovery, or Hot Reset. The initialization includes the Header Sequence Number Advertisement and the Rx Header Buffer Credit Advertisement between the two ports before a header packet can be transmitted.

- The following requirements shall be applied to a port:
  1. A port shall maintain two Tx Header Sequence Numbers. One is the Tx Header Sequence Number that is defined as the Header Sequence Number that will be assigned to a header packet when it is first transmitted (not a re-transmission). The other is the ACK Tx Header Sequence Number that is defined as the expected Header Sequence Number to be acknowledged with LGOOD_n that is sent by a port receiving the header packet.
  2. A port shall have an Rx Header Sequence Number. It is defined as the expected Header Sequence Number when a header packet is received.
  3. A port shall maintain two Rx Header Buffer Credit Counts. One is the Local Rx Header Buffer Credit Count that is defined as the number of the available Rx Header Buffer Credits of its receiver. The other is the Remote Rx Header Buffer Credit Count that is defined as the number of the available Rx Header Buffer Credits from its link partner.
  4. A port shall have enough Tx Header Buffers in its transmitter to hold up to four unacknowledged header packets.
  5. A port shall not transmit any header packet if its Remote Rx Header Buffer Credit Count is 0.

6. A port shall have enough Rx Header Buffers in its receiver to receive up to four header packets.
7. Upon entry to U0, the following shall be performed in the sequence presented:
   a. A port shall start the PENDING_HP_TIMER and CREDIT_HP_TIMER in expectation of the Header Sequence Number and the Rx Header Buffer Credit Advertisement.
   b. A port shall initiate the Header Sequence Number Advertisement.
   c. A port shall initiate the Rx Header Buffer Credit Advertisement.

- The Header Sequence Number Advertisement refers to ACK Tx Header Sequence Number initialization by exchanging Header Sequence Numbers between the two ports. This Header Sequence Number is the Header Sequence Number of the last header packet a port has received properly. The main purpose of the Header Sequence Number Advertisement is to maintain the link flow before and after Recovery such that a port upon re-entry to U0 is aware what the last header packet is that was sent successfully prior to Recovery, and decides what header packets in its Tx Header Buffers that can be flushed or need to be retransmitted. The following rules shall be applied during the Header Sequence Number Advertisement:
  1. A port shall set its initial Rx Header Sequence Number defined in the following:
     a. If a port enters U0 from Polling or Hot Reset, the Rx Header Sequence Number is 0.
     b. If a port enters U0 from Recovery, the Rx Header Sequence Number is the header Sequence Number of the next expected header packet.
  2. A port shall set its initial Tx Header Sequence Number defined in the following:
     a. If a port enters U0 from Polling or Hot Reset, its Tx Header Sequence Number is 0.
     b. If a port enters U0 from Recovery, its Tx Header Sequence Number is the same as the Tx Header Sequence Number before Recovery.

     Note: A header packet that is re-transmitted shall maintain its originally assigned Header Sequence Number.
  3. A port shall initiate the Header Sequence Number Advertisement by transmitting LGOOD_n with “n” equal to the Rx Header Sequence Number minus one.

     Note: The decrement is based on modulo-8 operation.
  4. A port shall set its initial ACK Tx Header Sequence Number to the Sequence Number received during the Rx Header Sequence Number Advertisement plus one.

     Note: The increment is based on modulo-8 operation.
  5. A port shall not send any header packets until the Header Sequence Number Advertisement has been received and a Remote Rx Header Buffer Credit is available.
  6. A port shall not request for a low power link state entry before receiving and sending the Header Sequence Number Advertisement.
     Note: The rules of Low Power Link State Initiation (refer to Section 7.2.4.2) still apply.

7. A port shall flush the header packets in its Tx Header Buffers upon receiving the Header Sequence Number Advertisement. A port shall do one of the following:
   a. If a port enters U0 from Polling or Hot Reset, it shall flush all the header packets in its Tx Header Buffers.
   b. If a port enters U0 from Recovery, it shall flush all the header packets in its Tx Header Buffers that have been sent before Recovery except for those with the Header Sequence Number greater than (modulo 8) the Header Sequence Number received in Header Sequence Number Advertisement.

   Note: If for example, the Header Sequence Number Advertisement of LGOOD_1 is received, a port shall flush the header packets in its Tx Header Buffers with Header Sequence Numbers of 1, 0, 7, 6.

- The Rx Header Buffer Credit Advertisement refers to Remote Rx Header Buffer Credit Count initialization by exchanging the number of available Local Rx Header Buffer Credits between the two ports. The main purpose of this advertisement is for a port to align its Remote Rx Header Buffer Credit Count with its link partner upon entry to U0. The following rules shall be applied during the Rx Header Buffer Credit Advertisement:
  1. A port shall initiate the Rx Header Buffer Credit Advertisement after sending LGOOD_n during Header Sequence Number Advertisement.
  2. A port shall initialize the following before sending the Rx Header Buffer Credit:
     a. A port shall initialize its Tx Header Buffer Credit index to A.
     b. A port shall initialize its Rx Header Buffer Credit index to A.
     c. A port shall initialize its Remote Rx Header Buffer Credit Count to 0.
     d. A port shall continue to process those header packets in its Rx Header Buffers that have been either acknowledged with LGOOD_n prior to entry to Recovery, or validated during Recovery, and then update the Local Rx Header Buffer Credit Count.
     e. A port shall set its Local Rx Header Buffer Credit Count defined in the following:
        1. If a port enters U0 from Polling or Hot Reset, its Local Rx Header Buffer Credit Count is 4.
        2. If a port enters U0 from Recovery, its Local Rx Header Buffer Credit Count is the number of Rx Header Buffers available for incoming header packets.
  3. A port shall perform the Rx Header Buffer Credit Advertisement by transmitting LCRD_x to notify its link partner. A port shall transmit one of the following based on its Local Rx Header Buffer Credit Count:
     a. LCRD_A if the Local Rx Header Buffer Credit Count is one.
     b. LCRD_A and LCRD_B if the Local Rx Header Buffer Credit Count is two.
     c. LCRD_A, LCRD_B, and LCRD_C if the Local Rx Header Buffer Credit Count is three.
     d. LCRD_A, LCRD_B, LCRD_C and LCRD_D if the Local Rx Header Buffer Credit Count is four.
  4. A port receiving LCRD_x from its link partner shall increment its Remote Rx Header Buffer Credit Count by one each time an LCRD_x is received up to four.
  5. A port shall not transmit any header packet if its Remote Rx Header Buffer Credit Count is zero.

6. A port shall not request for a low power link state entry before receiving and sending LCRD_x during the Rx Header Buffer Credit Advertisement.
   Note: The rules of Low Power Link State Initiation (refer to Section 7.2.4.2) still apply.

- The following rules shall be applied additionally when a port enters U0 from Recovery:
  1. A port sending LBAD before Recovery shall not expect to receive LRTY before a retried header packet from its link partner upon entry to U0.
  2. A port receiving LBAD before Recovery shall not send LRTY before a retried header packet to its link partner upon entry to U0.
     Note: There exists a situation where an LBAD was sent by a port before Recovery and it may or may not be received properly by its link partner. Under this situation, the rules of LBAD/LRTY do not apply. Refer to Sections 7.2.4.1.4 and 7.2.4.1.9 for details.
- Upon entry to Recovery and the next state is Hot Reset or Loopback, a port may optionally continue its processing of all the packets received properly.

7.2.4.1.2 General Rules of LGOOD_n and LCRD_x Usage

- The Rx Header Buffer Credit shall be transmitted in the alphabetical order of LCRD_A, LCRD_B, LCRD_C, LCRD_D, and back to LCRD_A. LCRD_x received out of alphabetical order is considered as missing of a link command, and transition to Recovery shall be initiated.
- Header packets shall be sent with the Header Sequence Number in the numerical order from 0 to 7, and back to 0. LGOOD_n received out of the numerical order is considered as missing of a link command, and the transition to Recovery shall be initiated.
- Header packet transmission may be delayed. When this occurs, the DL bit shall be set in the Link Control Word by a hub and optionally by a peripheral device or host. Some, but not necessarily all, of the conditions that will cause this delay follow:

  1. When a header packet is resent.
  2. When the link is in Recovery.
  3. When the Remote Rx Header Buffer Credit Count is zero.
  4. When the Tx Header Buffer is not empty.

  Note: The delayed bit only has significance if it is set in an ITP. If a device uses the ITP to synchronize its internal clock, then it should ignore any ITPs with the delayed bit set.

7.2.4.1.3 Transmitting Header Packets

- Before sending a header packet, a port shall add the Tx Header Sequence Number corresponding to the Header Sequence Number field in the Link Control Word.
- Transmission of a header packet shall consume a Tx Header Buffer. Accordingly, the Tx Header Sequence Number shall be incremented by one after the transmission or roll over to zero if the maximum Header sequence number is reached.
- Transmission of a retried header packet shall not consume an additional Tx Header Buffer and the Tx Header Sequence Number shall remain unchanged.
- Upon receiving LBAD, a port shall send LRTY followed by resending all the header packets that have not been acknowledged with LGOOD_n except for Recovery. Refer to Section 7.2.4.1.1 for additional rules applicable when a port enters U0 from Recovery.
- Prior to resending a header packet, a port shall set the Delay bit within the Link Control word and re-calculate CRC-5.

  Note: CRC-16 within header packet remains unchanged.

- The Remote Rx Header Buffer Credit Count shall be incremented by one if a valid LCRD_x is received.
- The Remote Rx Header Buffer Credit Count shall be decremented by one if a header packet is sent for the first time after entering U0, including when it is resent following Recovery.
- The Remote Rx Header Buffer Credit Count shall not be changed when a header packet is retried following LRTY.

7.2.4.1.4 Receiving Header Packets

- Upon receiving a header packet, the following verifications shall be performed:
  1. CRC-5
  2. CRC-16
  3. Matching between the Header Sequence Number in the received header packet and the Rx Header Sequence Number
  4. The availability of an Rx Header Buffer to store a header packet
- A header packet is defined as "received properly" when it has passed all four criteria described above.
- When a header packet has been received properly, a port shall issue a single LGOOD_n with "n" corresponding to the Rx Header Sequence Number and increment the Rx Header Sequence Number by one (or roll over to 0 if the maximum Header Sequence Number is reached).
- A port shall consume one Rx Header Buffer until it has been processed.
- When a header packet is not "received properly", one of the following shall occur:
  1. If the header packet has one or more CRC-5 or CRC-16 errors, a port shall issue a single LBAD. A port shall ignore all the header packets received subsequently until an LRTY has been received, or the link has entered Recovery. Refer to Section 7.2.4.1.1 for additional rules applicable when a port enters U0 from Recovery.
  2. If the Header Sequence Number in the received header packet does not match the Rx Header Sequence Number, or a port does not have an Rx Header Buffer available to store a header packet, a port shall transition to Recovery.
- After transmitting LBAD, a port shall continue to issue LCRD_x if an Rx Header Buffer Credit is made available.
- A port shall transition directly to Recovery if it fails to receive a header packet three consecutive times. A port shall not issue the third LBAD upon the third error.

7.2.4.1.5 Rx Header Buffer Credit

Each port is required to have four Rx Header Buffer Credits in its receiver. This is referred to the Local Rx Header Buffer Credit. The number of the Local Rx Header Buffer Credits represents the number of header packets a port can accept and is managed by the Local Rx Header Buffer Credit Count.

- A port shall consume one Local Rx Header Buffer Credit if a header packet is "received properly". The Local Rx Header Buffer Credit Count shall be decremented by one.
- Upon completion of a header packet processing, a port shall restore a Local Rx Header Buffer Credit by:
  1. Sending a single LCRD_x
  2. Advancing the Credit index alphabetically (or roll over to A if the Header Buffer Credit index of D is reached) and

3. Incrementing the Local Rx Header Buffer Credit Count by one.

Note: The LCRD_x index is used to ensure Rx Header Buffer Credits are sent in an alphabetical order such that missing of an LCRD_x can be detected.

7.2.4.1.6 Receiving Data Packet Payload

The processing of DPP shall adhere to the following rules:

- A DPP shall be accepted if the following two conditions are met:
  1. A DPH is received properly.
  2. A DPPStart ordered set is received properly immediately after its DPH.
- The DPP processing shall be completed when a valid DPPEND ordered set is detected.
- The DPP processing shall be aborted when one of the following conditions is met:
  1. A valid DPPABORT ordered set is detected.
  2. A K-symbol that does not belong to a valid DPPEND or DPPABORT ordered set is detected before a valid DPPEND or DPPABORT ordered set. A port shall then ignore the corresponding DPPEND or DPPABORT ordered set associated with the DPP.
  3. A DPP of length exceeding sDataSymbolsBabble (see Table 10-15) has been reached and no valid DPPEND or DPPABORT ordered set is detected.
- A DPP shall be dropped if its DPH is corrupted.
- A DPP shall be dropped when it does not immediately follow its DPH.

7.2.4.1.7 Receiving LGOOD_n

- A port shall maintain every header packet transmitted within its Tx Header Buffer until it receives an LGOOD_n. Upon receiving LGOOD_n, a port shall do one of the following:
  1. If LGOOD_n is the Header Sequence Number Advertisement and a port is entering U0 from Recovery, a port shall flush all the header packets retained in its Tx Header Buffers that have their Header Sequence Numbers equal to or less than the received Header Sequence Number, and initialize its ACK Tx Header Sequence Number to be the received Header Sequence Number plus one.

     Note: The comparison and increment are based on modulo-8 operation.
  2. If a port receives an LGOOD_n and this LGOOD_n is not Header Sequence Number Advertisement, it shall flush the header packet in its Tx Header Buffer with its Header Sequence Number matching the received Header Sequence Number and increment the ACK Tx Header Sequence Number by one based on modulo-8 operation.
  3. If a port receives an LGOOD_n and this LGOOD_n is not Header Sequence Number Advertisement, it shall transition to Recovery if the received Header Sequence Number does not match the ACK Tx Header Sequence Number. The ACK Tx Header Sequence Number shall be unchanged.

  Note: A port that has received an out of order LGOOD_n implies a lost or corrupted link command and shall initiate transition to Recovery.

7.2.4.1.8 Receiving LCRD_x

- A port shall adjust its Remote Rx Header Buffer Credit Count based on the received LCRD_x:
  1. A port shall increment its Remote Rx Header Buffer Credit Count by one upon receipt of LCRD_x.
  2. A port shall transition to Recovery if it receives an out of order LCRD_x.

  Note: A port that has received an out of order credit implies a lost or corrupted link command and shall transition to Recovery.

7.2.4.1.9 Receiving LBAD

- Upon receipt of LBAD, a port shall send a single LRTY before retransmitting all the header packets in the Tx Header Buffers that have not been acknowledged with LGOOD_n. A hub shall set the DL bit in the Link Control Word on all resent header packets and recalculate CRC-5. The host or a peripheral device may optionally set the DL bit in the Link Control Word on any resent header packets and recalculate CRC 5. If the retried packet is a DP and the DL bit in DPH is clear, the DPH must be followed by a DPP

  Note: Resending an ITP invalidates the isochronous timestamp value. CRC-16 is unchanged in a retried header packet.
- Upon receipt of LBAD, a port shall send a single LRTY if there is no unacknowledged header packet in the Tx Header Buffers.

  Note: This is an error condition where LBAD is created due to a link error.

7.2.4.1.10 Transmitter Timers

A PENDING_HP_TIMER is specified to cover the period of time from when a header packet is sent to a link partner, to when the header packet is acknowledged by a link partner. The purpose of this time limit is to allow a port to detect if the header packet acknowledgement sent by its link partner is lost or corrupted. The timeout value for the PENDING_HP_TIMER is listed in Table 7-7. The operation of the PENDING_HP_TIMER shall be based on the following rules:

- A port shall have a PENDING_HP_TIMER that is active only in U0 and if one of the following conditions is met:
  1. A port has a header packet transmitted but not acknowledged by its link partner, except during the period between receipt of LBAD and retransmission of the oldest header packet in the Tx Header Buffer.
  2. A port is expecting the Header Sequence Number Advertisement from its link partner.
- The PENDING_HP_TIMER shall be started if one of the following conditions is met:
  1. When a port enters U0 in expectation of the Header Sequence Number Advertisement.
  2. When a header packet is transmitted and there are no prior header packets transmitted but unacknowledged in the Tx Header Buffers.
  3. When the oldest header packet is retransmitted in response to LBAD.
- The PENDING_HP_TIMER shall be reset and restarted when a header packet is acknowledged with LGOOD_n and there are still header packets transmitted but unacknowledged in the Tx Header Buffers.

- The PENDING_HP_TIMER shall be reset and stopped if one of the following conditions is met:
  1. When a Header Sequence Number Advertisement is received.
  2. When a header packet acknowledgement of LGOOD_n is received and all the transmitted header packets in the Tx Header Buffers are acknowledged.
  3. When a header packet acknowledgement of LBAD is received.
- A port shall transition to Recovery if the following two conditions are met:
  1. PENDING_HP_TIMER times out.
  2. The transmission of an outgoing header packet is completed or the transmission of an outgoing DPP is either completed with DPPEND or terminated with DPPABORT.

  Note: This is to allow a graceful transition to Recovery without a header packet being truncated.

A CREDIT_HP_TIMER is also specified to cover the period of time from when a header packet has been transmitted and its Remote Rx Header Buffer Credit count is less than four, to when a Remote Rx Header Buffer Credit is received and its Remote Rx Header Buffer Credit count is back to four. The purpose of this timer is to make sure that a Remote Rx Header Buffer Credit is received within a reasonable time limit. This will allow a port sending the header packet to reclaim a Remote Rx Header Buffer Credit within a time limit in order to continue the process of packet transmission. This will also allow a port receiving the header packet enough time to process the header packet. The timeout value for the CREDIT_HP_TIMER is listed in Table 7-7. The operation of the CREDIT_HP_TIMER shall be based on the following rules:

- A port shall have a CREDIT_HP_TIMER that is active only in U0 and if one of the following conditions is met:
  1. A port has its Remote Rx Header Buffer Credit Count less than four.
  2. A port is expecting the Header Sequence Number Advertisement and the Rx Header Buffer Credit Advertisement from its link partner.
- The CREDIT_HP_TIMER shall be started when a header packet or a retried header packet is sent, or when a port enters U0.
- The CREDIT_HP_TIMER shall be reset when a valid LCRD_x is received.
- The CREDIT_HP_TIMER shall be restarted if a valid LCRD_x is received and the Remote Rx Header Buffer Credit Count is less than four.
- A port shall transition to Recovery if the following two conditions are met:
  1. CREDIT_HP_TIMER times out.
  2. The transmission of an outgoing header packet is completed or the transmission of an outgoing DPP is either completed with DPPEND or terminated with DPPABORT.

  Note: This is to allow a graceful transition to Recovery without a header packet being truncated.

Table 7-7. Transmitter Timers Summary

Timers	Timeout Value (µs)
PENDING_HP_TIMER	3
CREDIT_HP_TIMER	5000

7.2.4.2 Link Power Management and Flow

Requests to transition to low power link states are done at the link level during U0. Link commands LGO_U1, LGO_U2, and LGO_U3 are sent by a port as a request to enter a low power link state. LAU or LXU is sent by the other port as the response. LPMA is sent by a port in response only to LAU. Details on exit/wake from a low power link state are described in Sections 7.5.7, 7.5.8, and 7.5.9.

7.2.4.2.1 Power Management Link Timers

A port shall have three timers for link power management. First, a PM_LC_TIMER is used for a port initiating an entry request to a low power link state. It is designed to ensure a prompt entry to a low power link state. Second, a PM_ENTRY_TIMER is used for a port accepting the entry request to a low power link state. It is designed to ensure that both ports across the link are in the same low power link state regardless if the LAU or LPMA is lost or corrupted. Finally, a Ux_EXIT_TIMER is used for a port to initiate the exit from U1 or U2. It is specified to ensure that the duration of U1 or U2 exit is bounded and the latency of a header packet transmission is not compromised. The timeout values of the three timers are specified in Table 7-8.

A port shall operate the PM_LC_TIMER based on the following rules:

- A port requesting a low power link state entry shall start the PM_LC_TIMER after the last symbol of the LGO_Ux link command is sent.
- A port requesting a low power link state entry shall disable and reset the PM_LC_TIMER upon receipt of the LAU or LXU.

A port shall operate the PM_ENTRY_TIMER based on the following rules:

- A port accepting the request to enter a low power link state shall start the PM_ENTRY_TIMER after the last symbol of the LAU is sent.
- A port accepting the request to enter a low power link state shall disable and reset the PM_ENTRY_TIMER upon receipt of an LPMA or a TS1 ordered set.

A port shall operate the Ux_EXIT_TIMER based on the following rules.

- A port initiating U1 or U2 exit shall start the Ux_EXIT_TIMER when it starts to send LFPS Exit handshake signal.
- A port initiating U1 or U2 exit shall disable and reset the Ux_EXIT_TIMER upon entry to U0.

Table 7-8. Link Flow Control Timers Summary

Timers	Timeout Value (µs)
PM_LC_TIMER	3
PM_ENTRY_TIMER	6
Ux_EXIT_TIMER	6000

7.2.4.2.2 Low Power Link State Initiation

- A port shall not send a LGO_U1, LGO_U2 or LGO_U3 unless it meets all of the following:
  1. It has transmitted LGOOD_n and LCRD_x for all the header packets received.
  2. It has received LGOOD_n and LCRD_x for all the header packets transmitted.

  Note: This implies all credits must be received and returned before a port can initiate a transition to a low power link state.

  3. It has no pending packets for transmission.
  4. It has completed the Header Sequence Number Advertisement and the Rx Header Buffer Credit Advertisement upon entry to U0.

  Note: This implies that a port has sent the Header Sequence Number Advertisement and the Rx Header Buffer Credit Advertisement to its link partner, and also received the Header Sequence Number Advertisement and the Rx Header Buffer Credit Advertisement from its link partner.

  5. It is directed by a higher layer to initiate entry. Examples of when a higher layer may direct the link layer to initiate entry are: (a) the U1 or U2 inactivity timer expires (refer to PORT_U1_TIMEOUT, PORT_U2_TIMEOUT in Chapter 10); (b) reception of a SetPortFeature(PORT_LINK_STATE) request; and (c) device implementation specific mechanisms.
  6. It has met higher layer conditions for initiating entry. Examples are: (a) U1_enable/U2_enable is set or U1_TIMEOUT/U2_TIMEOUT is not equal zero; (b) device has received an ACK TP for each and every previously transmitted packet; (c) device is not waiting for a TP following a PING; and (d) device is not waiting for a timestamp following a timestamp request (for these and any other examples, refer to Chapter 8).
- A port shall do one of the following in response to receiving an LGO_U1 or LGO_U2:
  1. A port shall send an LAU if the Force Link PM Accept field is asserted due to having received a Set Link Functionality LMP.
  2. A port shall send an LAU if all of the following conditions are met:
     a. It has transmitted an LGOOD_n, LCRD_x sequence for all packets received.
     b. It has received an LGOOD_n, LCRD_x sequence for all packets transmitted.
     c. It has no pending packets for transmission.
     d. It is not directed by a higher layer to reject entry. Examples of when a higher layer may direct the link layer to reject entry are: (1) Downstream port is not enabled for U1 or U2 (i.e., PORT_U1_TIMEOUT or PORT_U2_TIMEOUT set to zero); (2) When a device has not received an ACK TP for a previously transmitted packet (refer to Chapter 8); and (3) When a device receives a ping TP (refer to the ping packet definition in Chapter 8 for more information).
  3. A port shall send an LXU if any of the above conditions are not met.

7.2.4.2.3 U1/U2 Entry Flow

Either a downstream port or an upstream port may initiate U1/U2 entry or exit. Entry to a low power U1 or U2 link state is accomplished by using the link commands defined in Table 7-5.

- A port shall send a single LGO_U1 or LGO_U2 to request a transition to a low power link state.
- Upon issuing LGO_Ux, a port shall start its PM_LC_TIMER.
- A port shall either accept LGO_Ux with a single LAU or shall reject LGO_U1 or LGO_U2 with a single LXU and remain in U0.
- Upon sending LGO_U1 or LGO_U2, a port shall not send any packets until it has received LXU or re-entered U0.
- Upon sending LGO_U1 or LGO_U2, a port shall continue receiving and processing packets and link commands.
- Upon receiving LXU, a port shall remain in U0.
- A port shall initiate transition to Recovery if a single LAU or LXU is not received upon PM_LC_TIMER timeout.
- Upon issuing LAU, a port shall start PM_ENTRY_TIMER.
- Upon receiving LAU, a port shall send a single LPMA and then enter the requested low power link state.
- Upon issuing LAU or LPMA, a port shall not send any packets or link commands.
- A port that sends LAU shall enter the corresponding low power link state upon receipt of LPMA before PM_ENTRY_TIMER timeout.
- A port that sends LAU shall enter the requested low power link state upon PM_ENTRY_TIMER timeout and if all of the following conditions are met:
  1. LPMA is not received.
  2. No TS1 ordered set is received.

  Note: This implies LPMA is corrupted and the port issuing LGO_Ux has entered Ux.
- A port that has sent LAU shall enter Recovery before PM_ENTRY_TIMER timeout if a TS1 ordered set is received.

  Note: This implies LAU was corrupted and the port issuing LGO_Ux has entered Recovery.
- A port that has sent LAU shall not respond with Ux LFPS exit handshake defined in Section 6.9.2 before PM_ENTRY_TIMER timeout and if LFPS Ux_Exit signal is received.

  Note: This implies LPMA was corrupted and the port issuing LGO_Ux has initiated Ux exit. Under this situation, the port sending LAU shall complete the low power link state entry process and then respond to Ux exit.

There also exists a situation where a port transitions from U1 to U2 directly.

- A port in U1 shall enter U2 directly if the following two conditions are met:
  1. The port's U2 inactivity timer is enabled.
  2. The U2 inactivity timer times out and no U1 LFPS exit signal is received.

7.2.4.2.4 U3 Entry Flow

Only a downstream port can initiate U3 entry. An upstream port shall not reject U3 entry.

- Upon directed, a downstream port shall initiate U3 entry process by sending LGO_U3.
- Upon issuing LGO_U3, a downstream port shall start PM_LC_TIMER.
- An upstream port shall send LAU in response to LGO_U3 request by a downstream port.
- An upstream port shall not send any packets or link commands subsequent to sending an LAU.
- Upon issuing LGO_U3, a downstream port shall ignore any packets sent by an upstream port.

  Note: This is a corner condition that an upstream port is sending a header packet before receiving LGO_U3.
- Upon Receiving LGO_U3, an upstream port shall respond with an LAU. The processing of all the unacknowledged packets shall be aborted.
- Upon issuing LAU, an upstream port shall start PM_ENTRY_TIMER.
- A downstream port shall send a single LPMA and then transition to U3 when LAU is received.
- A downstream port shall transition to Recovery and reinitiate U3 entry after re-entry to U0 if all of the following three conditions are met:
  1. The PM_LC_TIMER times out.
  2. LAU is not received.
  3. The number of consecutive U3 entry attempts is less than three.
- An upstream port shall transition U3 when one of the following two conditions is met:
  1. LPMA is received
  2. The PM_ENTRY_TIMER times out and LPMA is not received
- A downstream port shall transition to SS.Inactive when it fails U3 entry on three consecutive attempts.

7.2.4.2.5 Concurrent Low Power Link Management Flow

Concurrent low power link management flow applies to situations where a downstream port and an upstream port both issue a request to enter a low power link state.

- If a downstream port has sent an LGO_U1, LGO_U2, or LGO_U3 and also received an LGO_U1 or LGO_U2, it shall send an LXU.
- If an upstream port has sent an LGO_U1 or LGO_U2 and also received an LGO_U1, LGO_U2, it shall wait until receipt of an LXU and then send either an LAU or LXU.
- If an upstream port has sent an LGO_U1 or LGO_U2 and also received an LGO_U3 from a downstream port, it shall wait until the reception of an LXU and then send an LAU.
- If a downstream port is directed by a higher layer to initiate a transition to U3, and a transition to U1 or U2 has been initiated but not yet completed, the port shall first complete the in-process transition to U1 or U2, then return to U0 and request entry to U3.

7.2.4.2.6 Concurrent Low Power Link Management and Recovery Flow

Concurrent low power link management and Recovery flow applies to situations where a port issues a low power link state entry and another port issues Recovery. The port that issues the low power link state entry shall meet the following rules:

- Upon issuing LGO_Ux, the port shall transition to Recovery if a TS1 ordered set is received.
- The port shall reinitiate low power link state entry process described in Section 7.2.4.2.3 and 7.2.4.2.4 upon re-entry to U0 from Recovery if the conditions to enter a low power link state are still valid.

7.2.4.2.7 Low Power Link State Exit Flow

Exit from a low power link state refers to exit from U1/U2, or wakeup from U3. It is accomplished by the LFPS Exit signaling defined in Section 6.9.2. A successful LFPS handshake process will lead both a downstream port and an upstream port to Recovery.

A Ux_EXIT_TIMER defined in Section 7.2.4.2.1 is only applied when a port is attempting an exit from U1 or U2. It shall not be applied when a port is initiating a U3 wakeup.

The exit from U1/U2 shall meet the following flow. The U3 wakeup follows the same flow with the exception that Ux_EXIT_TIMER is disabled during U3 wakeup.

- If a port is initiating U1/U2 Exit, it shall start sending U1/U2 LFPS Exit handshake signal defined in Section 6.9.2 and start the Ux_EXIT_TIMER.
- If a port is initiating U3 wakeup, it shall start sending U3 LFPS wakeup handshake signal defined in Section 6.9.2.
- A port upon receiving U1/U2 Exit or U3 wakeup LFPS handshake signal shall start U1/U2 exit or U3 wakeup by responding with U1/U2 Exit or U3 wakeup LFPS signal defined in Section 6.9.2.
- Upon a successful LFPS handshake before tNoLFPSResponseTimeout defined in Table 6-14, a port shall transition to Recovery.
- A port initiating U1 or U2 Exit shall transition to SS.Inactive if one of the following two conditions is met:
  1. Upon tNoLFPSResponseTimeout and the condition of a successful LFPS handshake is not met.
  2. Upon Ux_EXIT_TIMER timeout, the link has not transitioned to U0.
- A port initiating U3 wakeup shall remain in U3 when the condition of a successful LFPS handshake is not met upon tNoLFPSResponseTimeout and it may initiate U3 wakeup again after a minimum of 100-ms delay.
- A root port not able to respond to U3 LFPS wakeup within tNoLFPSResponseTimeout shall initiate U3 LFPS wakeup when it is ready to return to U0.

7.3 Link Error Rules/Recovery

7.3.1 Overview of SuperSpeed Bit Errors

The SuperSpeed timing budget is based on a link's statistical random bit error probability less than 10^{-12}. Packet framings and link command framing are tolerant to one symbol error. Details on bit error detection under link flow control are described in Section 7.2.4.

7.3.2 Link Error Types, Detection, and Recovery

Data transfers between the two link partners are carried out using the form of a packet. A set of link commands is defined to ensure the successful packet flow across the link. Other link commands are also defined to manage the link connectivity. When symbol errors occur on the link, the integrity of a packet or a link command can be compromised. Therefore, not only a packet or a link command needs to be constructed to increase the error tolerance, but the link data integrity handling also needs to be specified such that any errors that will invalidate or corrupt a packet or a link command can be detected and a link error can be recovered.

There are various types of errors at the link layer. This includes an error on a packet or a link command, or an error during the link training process, or an error when a link is in transition from one state to another. The detection and recovery from those link errors are described with details in this section.

7.3.3 Header Packet Errors

Several types of header packet errors are detected. They are:

1. Missing of a header packet
2. Invalid header packet due to CRC errors
3. Mismatch of a Rx Header Sequence Number

Regardless, the Link Error Count is incremented for only one class of errors in the link layer, and those are errors which will cause the link to transition to Recovery. For errors that will not cause the link to enter Recovery, the Link Error Count shall remain unchanged.

7.3.3.1 Packet Framing Error

A packet framing ordered set is constructed such that any single K-symbol corruption within the ordered set will not prevent its packet framing recognition.

Header packet framing and DPP framing are all constructed using four K-symbol ordered sets. A header packet contains only one packet framing ordered set at the beginning of the packet defined in Section 7.2.1. A DPP begins with start packet framing ordered set and ends with end packet framing ordered set as defined in Section 7.2.2.

- A valid HPSTART ordered set shall be declared if the following two conditions are met:
  1. At least three of the four K-symbols in the four consecutive symbol periods are valid packet Framing K-symbols.
  2. The four symbols are in the order defined in Table 7-9.

Note: If an HPSTART ordered set has two or more K-symbols corrupted, a header packet will not be detectable and, therefore, result in missing of a header packet.

- Missing of a header packet shall result in a port transitioning to Recovery depending on which one of the following conditions becomes true first:
  1. A port transmitting the header packet upon its PENDING_HP_TIMER timeout.
  2. A port receiving the header packet upon detection of a Rx Header Sequence Number error.
- The Link Error Count shall be incremented by one each time a transition to Recovery occurs.

Table 7-9. Valid Packet Framing K-Symbol Order (K is One of SHP, SDP, END or EDB)

Symbol 0	Symbol 1	Symbol 2	Symbol 3	Comment
K	K	K	EPF	All K-symbols are valid
Corrupt	K	K	EPF	First K corrupted
K	Corrupt	K	EPF	Second K corrupted
K	K	Corrupt	EPF	Third K corrupted
K	K	K	Corrupt	EPF corrupted

7.3.3.2 Header Packet Error

Each header packet contains a CRC-5 and a CRC-16 to ensure that the data integrity of a header packet can be verified. A CRC-5 is used to detect bit errors in the Link Control Word. A CRC-16 is used to detect bit errors in the packet header. A header packet error can be detected using CRC-5 or CRC-16 checks.

- A header packet error shall be declared if the following conditions are true:
  1. A valid HPSTART ordered set is detected.
  2. Either CRC-5 or CRC-16 check fails as defined in Section 7.2.1 or any K-symbol occurrence in the packet header or Link Control Word that prevents CRC-5 or CRC-16 checks from being completed.
- A port receiving the header packet shall send an LBAD as defined in Section 7.2.4.1 if it detects a header packet error. The Link Error Count shall remain unchanged.
- If a port fails to receive a header packet for three consecutive times, it shall transition to Recovery. The Link Error Count shall be incremented by one. Refer to Section 7.2.4.1.4 for details.

7.3.3.3 Rx Header Sequence Number Error

Each port contains an Rx Header Sequence Number that is defined in Section 7.2.4.1 and initialized upon entry to U0. Upon receiving a header packet, a port is required to compare the Header Sequence Number embedded in the header packet with the Rx Header Sequence Number stored in its receiver. This ensures that header packets are transmitted and received in an orderly manner. A missing or corrupted header packet can be detected.

- An Rx Header Sequence Number error shall occur if the following conditions are met:
  1. A header packet is received and no header packet error is detected.
  2. The Header Sequence Number in the received header packet does not match the Rx Header Sequence Number.
- A port detecting an Rx Header Packet Sequence Number error shall transition to Recovery.
- The Link Error Count shall be incremented by one each time a transition to Recovery occurs.

7.3.4 Link Command Errors

A link command consists of four K-symbol link command frame ordered set, LCSTART, followed by a two-symbol link command word, and its repeat. A link command is constructed such that any single K-symbol corruption within the link command frame ordered set will not invalidate the recognition of a link command, and any single-bit error in the two scrambled link command words will not corrupt the correct parsing of a link command.

- A detection of a link command shall be declared if the following two conditions are met:
  1. At least three of the four K-symbols in four consecutive symbol periods are valid link command K-symbols.
  2. The four symbols are in the order described in Table 7-10.
- A valid link command is declared if both link command words are the same, they both contain valid link command information as defined in Table 7-4, and they both pass the CRC-5 check.
- An invalid link command is declared upon detection of a link command and the conditions to meet a valid link command are not met.
- An invalid link command shall be ignored.
- A port detecting missing of LGOOD_n or LCRD_x shall transition to Recovery.

  Note: Missing LGOOD_n is declared when two consecutive LGOOD_n received are not in numerical order. Missing LGOOD_n, or LBAD, or LRTY can also be inferred upon PENDING_HP_TIMER timeout. Missing LCRD_x is declared when two consecutive LCRD_x received are not in alphabetical order, or upon CREDIT_HP_TIMER times out and LCRD_x is not received.
- A port detecting missing of LGO_Ux, or LAU, or LXU shall transition to Recovery.

  Note: Detection of missing LGO_Ux, or LAU, or LXU is declared upon PM_LC_TIMER timeout and LAU or LXU is not received.
- A downstream port detecting missing of LUP shall transition to Recovery (refer to Section 7.5.6 for LUP detection).

  Note: Missing of LPMA will not transition the link to Recovery. It will only cause an Ux entry delay for the port accepting LGO_Ux (refer to Section 7.2.4.2 for details).

- An upstream port detecting missing of LDN shall transition to Recovery (refer to Section 7.5.6 for LDN detection).
- The Link Error Count shall be incremented by one each time a transition to Recovery occurs due to an error.

Table 7-10. Valid Link Command K-Symbol Order

Symbol 0	Symbol 1	Symbol 2	Symbol 3	Comment
SLC	SLC	SLC	EPF	All K-symbols are valid
Corrupt	SLC	SLC	EPF	First SLC corrupted
SLC	Corrupt	SLC	EPF	Second SLC corrupted
SLC	SLC	Corrupt	EPF	Third SLC corrupted
SLC	SLC	SLC	Corrupt	EPF corrupted

7.3.5 ACK Tx Header Sequence Number Error

Each port has an ACK Tx Header Sequence Number that is defined in Section 7.2.4.1. The ACK Tx Header Sequence Number is initialized during the Header Sequence Number Advertisement. After a header packet is transmitted, a port is expecting to receive an LGOOD_n from its link partner as an explicit acknowledgement that the header packet is received properly. Upon receiving LGOOD_n, the Header Sequence Number contained in LGOOD_n will be compared with the ACK Tx Header Sequence Number. The outcome of the comparison will determine if an ACK Tx Header Sequence Number error has occurred.

- An ACK Tx Header Sequence Number error shall be declared if the following conditions are met:
  1. A valid LGOOD_n is received.
  2. The Header Sequence Number in the received LGOOD_n does not match the ACK Tx Header Sequence Number.
  3. The LGOOD_n is not for Header Sequence Number Advertisement.
- A port detecting an ACK Tx Header Sequence Number error shall transition to Recovery.
- The Link Error Count shall be incremented by one each time a transition to Recovery occurs.

7.3.6 Header Sequence Number Advertisement Error

Each port is required to first perform a Header Sequence Number Advertisement upon entry to U0. The details of a Header Sequence Number Advertisement are described in Section 7.2.4. A Header Sequence Number Advertisement is the first step of the link initialization to ensure that the link flow is maintained un-interrupted before and after Recovery. Any errors occurred during the Header Sequence Number Advertisement must be detected and proper error recovery must be initiated.

- A Header Sequence Number Advertisement error shall occur if one of the following conditions is true:
  1. Upon PENDING_HP_TIMER timeout and the Header Sequence Number Advertisement not received
  2. A header packet received before sending Header Sequence Number Advertisement
  3. LCRD_x or LGO_Ux received before receiving Header Sequence Number Advertisement

- A port detecting any Header Sequence Number Advertisement error shall transition to Recovery.
- The Link Error Count shall be incremented by one each time a transition to Recovery occurs.

7.3.7 Rx Header Buffer Credit Advertisement Error

Each port is required to perform the Rx Header Buffer Credit Advertisement after Header Sequence Number Advertisement upon entry to U0. The details of Rx Header Buffer Credit Advertisement are described in Section 7.2.4.

- An Rx Header Buffer Credit Advertisement error shall occur if one of the following conditions is true:
  1. Upon CREDIT_HP_TIMER timeout and no LCRD_x received.
  2. A header packet received before sending LCRD_x.
  3. LGO_Ux received before receiving LCRD_x.
- A port detecting an Rx Header Buffer Credit Advertisement Error shall transition to Recovery.
- The Link Error Count shall be incremented by one each time a transition to Recovery occurs.

7.3.8 Training Sequence Error

Symbol corruptions during the TS1 and TS2 ordered sets in Polling.Active, Polling.Configuration, Recovery.Active, and Recovery.Configuration substates are expected until the requirements are met to transition to the next state. A timeout from any one of these substates is considered Training Sequence error.

- A timeout from either Polling.Active, Polling.Configuration, Recovery.Active, or Recovery.Configuration substate shall result in a Training Sequence error.
- Upon detecting a Training Sequence error, one of the following link state transitions shall be followed:
  1. A downstream port shall transition to Rx.Detect if a Training Sequence error occurs during Polling.
  2. An upstream port of a hub shall transition to Rx.Detect if a Training Sequence error occurs during Polling.
  3. An upstream port of a peripheral device shall transition to SS.Disabled if a Training Sequence error occurs during Polling.
  4. A downstream port shall transition to SS.Inactive if a Training Sequence error occurs during Recovery and the transition to Recovery is not an attempt for Hot Reset.
  5. A downstream port shall transition to Rx.Detect if a Training Sequence error occurs during Recovery.Active and Recovery.Configuration and the transition to Recovery is an attempt for Hot Reset.
  6. An upstream port shall transition to SS.Inactive if a Training Sequence error occurs during Recovery.
- The Link Error Count shall remain unchanged.

7.3.9 8b/10b Errors

There are two types of errors when a receiver decodes 8b/10b symbols. One is a disparity error that is declared when the running disparity of the received 8b/10b symbols is not +2, or 0, or -2. The other is a decode error when an unrecognized 8b/10b symbol is received.

Upon receiving notification of an 8b/10b error:

- A port may optionally do the following:
  1. If the link is receiving a header packet, it shall send LBAD.
  2. If the link is receiving a link command, it shall ignore the link command.
  3. If the link is receiving a DPP, it shall drop the DPP.
- The Link Error Count shall remain unchanged.

7.3.10 Summary of Error Types and Recovery

Table 7-11 summarizes the link error types, error count, and different error paths to restore the link.

- The link error shall be counted each time a link transitions to Recovery due to an error.
- The link error shall be counted by a downstream port.
- The Link Error Count shall be reset upon PowerOn Reset, Warm Reset, Hot Reset, or whenever a port enters Polling.Idle.

Situations also exist where an unexpected link command or header packet is received. These include but are not limited to the following:

1. Receiving an unexpected link command such as LBAD, LRTY, LAU, LXU, or LPMA before receiving the Header Sequence Number Advertisement and the Remote Rx Header Buffer Credit Advertisement.
2. Receiving the Header Sequence Number Advertisement after entry to U0 from Recovery with its ACK Tx Header Sequence Number not corresponding to any header packets in the Tx Header Buffers.
3. Receiving LRTY without sending LBAD.
4. Receiving LGOOD_n that is neither a Header Sequence Number Advertisement, nor for header packet acknowledgement.
5. Receiving LAU, or LXU without sending LGO_Ux.
6. Receiving LPMA without sending LAU.
7. Receiving an unexpected header packet during link initialization.

These error situations are largely not due to link errors. A port's behavior under these situations is undefined and implementation specific. It is recommended that a port ignore those unexpected link commands or header packets.

If the ports are directed to different link states based on TS2 ordered set, the downstream port's TS2 ordered set overrides the upstream port's. For example, if a downstream port issues Hot Reset in its TS2 ordered set, and an upstream port issues Loopback mode, Hot Reset overrides Loopback. The ports shall enter Hot Reset.

Table 7-11. Error Types and Recovery

Error Type	Description/Example	Error Recovery Path	Update Link Error Count?
Missing Header Packet Framing	Only a valid packet framing ordered set will be declared in the receiver side.	Delayed transition to Recovery	Yes
Header Packet Error	Any header packet CRC is bad.	Header packet retry process	No
Rx Header Sequence Number Error	The Header Sequence Number in the received header packet does not match the Rx Header Sequence Number.	Recovery	Yes
ACK Tx Header Sequence Number Error	The Header Sequence Number in the received LGOOD_n (not Header Sequence Number Advertisement) does not match ACK Tx Header Sequence Number.	Recovery	Yes
Header Sequence Number Advertisement Error	1. LGOOD_n not received upon PENDING_HP_TIMER timeout. 2. A header packet received before sending LGOOD_n. 3. LCRD_x or LGO_Ux received before receiving LGOOD_n.	Recovery	Yes
Rx Header Buffer Credit Advertisement Error	1. LCRD_x not received upon CREDIT_HP_TIMER timeout. 2. A header packet received before sending LCRD_x. 3. LGO_Ux received before receiving LCRD_x.	Recovery	Yes
Training Sequence Error	1. Timeout from Polling to Rx.Detect or SS.Disabled without reaching U0. 2. Timeout from Recovery to SS.Inactive without reaching U0.	Timeout from Recovery to SS.Inactive requires software intervention.	No
Invalid link command	Valid link command framing but invalid link command word.	Ignored	No
Missing link command	No valid link command framing is detected.	Delayed transition to Recovery if missing LGOOD_n or LCRD_x	Yes
8b/10b Error	Detected in the PHY layer	N.A.	No

7.4 PowerOn Reset and Inband Reset

There are two categories of reset associated with a link. The first, PowerOn Reset, restores storage elements, registers, or memories to predetermined states when power is applied. Upon PowerOn Reset, the LTSSM (described in Section 7.5) shall enter Rx.Detect. The second, Inband Reset, uses SuperSpeed or LFPS signaling to propagate the reset across the link. There are two mechanisms to complete an Inband Reset, Hot Reset and Warm Reset. Upon completion of either a PowerOn Reset or an Inband Reset, the link shall transition to U0 as described in Section 7.4.2.

7.4.1 PowerOn Reset

PowerOn Reset restores a storage element, register, or memory to a predetermined state when power is applied (refer to Section 9.1.1.2 for clarification of when power is applied for self powered devices). A port must be responsible for its own internal Reset signaling and timing.

The following shall occur when PowerOn Reset is asserted or while VBUS is off:

1. Receiver termination shall meet the $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}POS}$ specification defined in Table 6-13.
2. Transmitters shall hold a constant DC common mode voltage ($V_{TX\text{-}DC\text{-}CM}$) defined in Table 6-11.

The following shall occur when PowerOn Reset is completed and VBUS is valid:

1. The LTSSM of a port shall be initialized to Rx.Detect.
2. The LTSSM and the PHY level variables (such as Rx equalization settings) shall be reset to their default values.
3. The receiver termination of a port shall meet the $R_{RX\text{-}DC}$ specification defined in Table 6-13.

   Note: Rx termination shall always be maintained throughout operation except for SS.Disabled

7.4.2 Inband Reset

An Inband Reset shall be generated by a downstream port only when it is directed.

There are two mechanisms to generate an Inband Reset. The first mechanism; Hot Reset, is defined by sending TS2 ordered sets with the Reset bit asserted. A Hot Reset shall cause the LTSSM to transition to the Hot Reset state. Upon completion of Hot Reset, the following shall occur:

- A downstream port shall reset its Link Error Count.
- The port configuration information of an upstream port shall remain unchanged. Refer to Sections 8.4.5 and 8.4.6 for details.
- The PHY level variables (such as Rx equalization settings) shall remain unchanged.
- The LTSSM of a port shall transition to U0.

The second mechanism of an Inband Reset is Warm Reset. The signaling of a Warm Reset is defined as an LFPS signaling meeting the tReset requirements (see Table 6-20). A Warm Reset will cause the LTSSM to transition to Rx.Detect, retrain the link including the receiver equalizer, reset an upstream port, and then transition to U0. An upstream port shall enable its LFPS receiver and Warm Reset detector in all link states except SS.Disabled. A completion of a Warm Reset shall result in the following.

- A downstream port shall reset its Link Error Count.
- Port configuration information of an upstream port shall be reset to default values. Refer to Sections 8.4.5 and 8.4.6 for details.
- The PHY level variables (such as Rx equalization settings) shall be reinitialized or retrained.
- The LTSSM of a port shall transition to U0 through RxDetect and Polling.

A downstream port may be directed to reset the link in two ways, “PORT_RESET”, or “BH_PORT_RESET” as described in Section 10.3.1.6. When a “PORT_RESET” is directed, a downstream port shall issue either a Hot Reset, or a Warm Reset, depending on its LTSSM state. When a “BH_PORT_RESET” is directed, a downstream port shall issue a Warm Reset in any of its LTSSM states except SS.Disabled.

If a "PORT_RESET" is directed, a downstream port shall issue either a Hot Reset or a Warm Reset based on the following conditions:

- If the downstream port is U3, or Loopback, or Compliance Mode, or SS.Inactive, it shall use Warm Reset.
- If the downstream port is in U0, it shall use Hot Reset.
- If a downstream port is in a transitory state of Polling or Recovery, it shall use Hot Reset.
- If the downstream port is in U1 or U2, it shall exit U1 or U2 using the LFPS exit handshake, transition to Recovery and then transition to Hot Reset. The following two additional rules apply when the downstream port fails to enter Hot Reset.
  1. If a Hot Reset fails due to an LFPS handshake timeout in U1 or U2, a downstream port shall transition to SS.Inactive until software intervention or upon detection of removal of an upstream port.
  2. If a Hot Reset fails due to a TS1/TS2 handshake timeout, a downstream port shall transition to Rx.Detect and attempt a Warm Reset.
- If the downstream port is in SS.Disabled, an Inband Reset is prohibited.

If a "BH_PORT_RESET" is directed, Warm Reset shall be issued, and the following shall occur:

- A downstream port shall initiate a Warm Reset in all the link states except SS.Disabled and transition to Rx.Detect.
- An upstream port shall enable its LFPS receiver and Warm Reset detector in all the link states except SS.Disabled.
- An upstream port receiving Warm Reset shall transition to Rx.Detect. Refer to Section 6.9.3 for Warm Reset Detection.

7.5 Link Training and Status State Machine (LTSSM)

Link Training and Status State Machine (LTSSM) is a state machine defined for link connectivity and the link power management. LTSSM consists of 12 different link states that can be characterized based on their functionalities. First, there are four operational link states, U0, U1, U2, and U3. U0 is a state where a SuperSpeed link is enabled. Packet transfers are in progress or the link is idle. U1 is low power link state where no packet transfer is carried out and the SuperSpeed link connectivity can be disabled to allow opportunities for saving the link power. U2 is also a low power link state. Compared with U1, U2 allows for further power saving opportunities with a penalty of increased exit latency. U3 is a link suspend state where aggressive power saving opportunities are possible.

Second, there are four link states, Rx.Detect, Polling, Recovery, and Hot Reset, that are introduced for link initialization and training. Rx.Detect represents the initial power-on link state where a port is attempting to determine if its SuperSpeed link partner is present. Upon detecting the presence of a SuperSpeed link partner, the link training process will be started. Polling is a link state that is defined for the two link partners to have their SuperSpeed transmitters and receivers trained, synchronized, and ready for packet transfer. Recovery is a link state defined for retraining the link when the two link partners exit from a low power link state, or when a link partner has detected that the link is not operating in U0 properly and the link needs to be retrained, or when a link partner decides to change the mode of link operation. Hot Reset is a state defined to allow a downstream port to reset its upstream port.

Third, two other link states, Loopback and Compliance Mode, are introduced for bit error test and transmitter compliance test. Finally, two more link states are defined. SS.Inactive is a link error state where a link is in a non-operable state and software intervention is needed. SS.Disabled is a link state where SuperSpeed connectivity is disabled and the link may operate under USB 2.0 mode.

Configuration information and requests to initiate LTSSM state transitions are mainly controlled by software. All LTSSM references to "directed" refers to upper layer mechanisms.

There are also various timers defined and implemented for LTSSM in order to ensure the successful operation of LTSSM. The timeout values are summarized in Table 7-12. All timers used in the link layer have a tolerance of 0~+50% accuracy with exception of the U2 inactivity timer (refer to Section 10.4.1 for U2 inactivity timer accuracy). All timeout values must be set to the specified values after PowerOn Reset or Inband Reset. All counters must be also initialized after PowerOn Reset or Inband Reset.

In the state machine descriptions, lists of state entry and exit conditions are not prioritized.

State machine diagrams are overviews and may not include all the transition conditions.

Table 7-12. LTSSM State Transition Timeouts

Name	Initial State	Timeout to the Next State	Timeout Value
tSSInactiveQuietTimeout	SS.Inactive.Quiet	SS.Inactive.Disconnect.Detect	12 ms
tRxDetectQuietTimeout	Rx.Detect.Quiet	Rx.Detect.Active	12 ms
tPollingLFPSTimeout	Polling.LFPS [1]	Compliance/Rx.Detect/SS.Disabled	360 ms
tPollingActiveTimeout	Polling.Active[1]	Rx.Detect/SS.Disabled	12 ms
tPollingConfigurationTimeout	Polling.Configuration[1]	Rx.Detect/SS.Disabled	12 ms
tPollingIdleTimeout	Polling.Idle[1]	Rx.Detect/SS.Disabled	2 ms
tU0RecoveryTimeout	U0	Recovery	1 ms
tU0LTimeout	U0	U0	10 µs
tNoLFPSResponseTimeout	U1	SS.Inactive	2 ms
PORT_U2_TIMEOUT	U1[2]	U2	U2 Inactivity field set in LMP (refer to Section 8.4 for details)
tU1PingTimeout	U1	Rx.Detect	300 ms
tNoLFPSResponseTimeout	U2	SS.Inactive	2 ms
tNoLFPSResponseTimeout	U3	U3	10 ms
tRecoveryActiveTimeout	Recovery.Active	SS.Inactive, Rx.Detect	12 ms
tRecoveryConfigurationTimeout	Recovery.Configuration	SS.Inactive, Rx.Detect	6 ms
tRecoveryIdleTimeout	Recovery.Idle	SS.Inactive	2 ms
tLoopbackExitTimeout	Loopback.Exit	SS.Inactive	2 ms

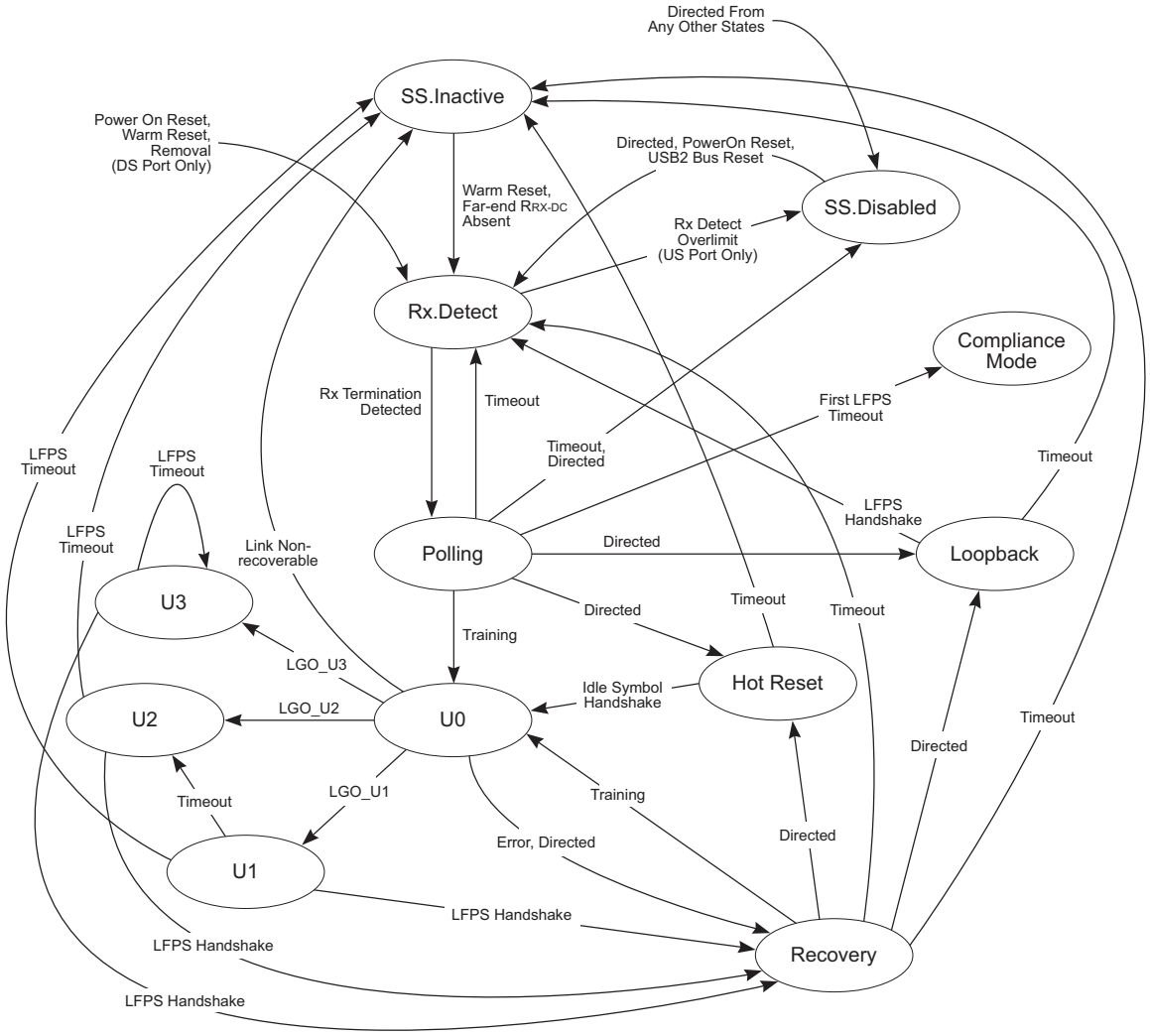
Name	Initial State	Timeout to the Next State	Timeout Value
tSSInactiveQuietTimeout	SS.Inactive.Quiet	SS.Inactive.Disconnect.Detect	12 ms
tRxDetectQuietTimeout	Rx.Detect.Quiet	Rx.Detect.Active	12 ms
tPollingLFPSTimeout	Polling.LFPS [1]	Compliance/Rx.Detect/SS.Disabled	360 ms
tPollingActiveTimeout	Polling.Active[1]	Rx.Detect/SS.Disabled	12 ms
tPollingConfigurationTimeout	Polling.Configuration[1]	Rx.Detect/SS.Disabled	12 ms
tPollingIdleTimeout	Polling.Idle[1]	Rx.Detect/SS.Disabled	2 ms
tU0RecoveryTimeout	U0	Recovery	1 ms
tHotResetActiveTimeout	Hot Reset.Active	SS.Inactive	12 ms
tHotResetExitTimeout	Hot Reset.Exit	SS.Inactive	2 ms
tU3WakeupRetryDelay	U3	U3	100 ms
tU2RxdetDelay	U2	U2	100 ms
tU3RxdetDelay	U3	U3	100 ms

Notes:

1. Upon Polling timeout, a port shall transition to different states. Refer to Section 7.5.4.3 for details.

2. The accuracy of U2 inactivity timer is specified in Section 10.4.1.

All state machines diagrams have descriptions for transition conditions. These descriptions are informative only. The exact implementation of the state transitions shall follow the description in each section.



Note: Transition conditions are illustrative only. Not all of the transition conditions are listed.

U-047A

Figure 7-13. State Diagram of the Link Training and Status State Machine

7.5.1 SS.Disabled

SS.Disabled is a state with a port's low-impedance receiver termination removed. It is a state where a port's SuperSpeed connectivity is disabled. Refer to Section 10.16 for details regarding the behavior of a peripheral device. Refer to Sections 10.3 to 10.6 for behaviors regarding a hub's upstream port and downstream port.

SS.Disabled does not contain any substates in the case of downstream ports and hub upstream ports. For a peripheral upstream port, it contains two substates, SS.Disabled.Default and SS.Disabled.Error.

SS.Disabled is also a logical power-off state for a self-powered hub upstream port.

SS.Disabled.Default is also a logical power-off state for a self-powered peripheral upstream port.

A downstream port shall transition to this state from any other state when directed.

A self-powered hub or peripheral upstream port shall transition to this state when VBUS is not valid.

7.5.1.1 SS.Disabled for Downstream Ports and Hub Upstream Ports

SS.Disabled for Downstream Ports and Hub Upstream Ports does not contain any substates.

7.5.1.1.1 SS.Disabled Requirements

- VBUS may be present during SS.Disabled.
- The port's receiver termination shall present high impedance to ground of $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}POS}$ defined in Table 6-13.
- The port shall be disabled from transmitting and receiving LFPS and SuperSpeed signals.

7.5.1.1.2 Exit from SS.Disabled

- A downstream port shall transition to Rx.Detect when directed.
- An upstream port shall transition to Rx.Detect only when VBUS transitions to valid or a USB 2.0 bus reset is detected.

7.5.1.2 SS.Disabled for Upstream Ports of Peripheral Devices

SS.Disabled of a peripheral device operates similarly to hub upstream ports, except that it only attempts a limited number of SuperSpeed attempts upon USB 2.0 bus reset.

7.5.1.2.1 SS.Disabled Substate Machine

SS.Disabled of a peripheral device has two substates shown in Figure 7-14.

- SS.Disabled.Default
- SS.Disabled.Error

SS.Disable.Default is a logical power-off state for a self-powered peripheral device.

7.5.1.2.2 SS.Disabled Requirements

The requirements of a peripheral upstream port are the same as defined in Section 7.5.1.1.1. In addition, a peripheral upstream port shall implement a tDisabledCount counter. The operation of the tDisabledCount counter shall meet the following requirement.

- The tDisabledCount counter shall be reset to zero upon one of the following two conditions:
  1. Invalid VBUS
  2. Successful port configuration exchange
- The tDisabledCount counter shall be incremented upon entry to SS.Disabled.Default.

7.5.1.2.3 Exit from SS.Disabled.Default

- A peripheral upstream port shall transition to Rx.Detect if one of the following conditions are met:
  1. When VBUS transitions to valid.
  2. When a USB 2.0 bus reset is detected and tDisabledCount is less than 3.
- A peripheral upstream port shall transition to SS.Disabled.Error if tDisabledCount is 3.

7.5.1.2.4 Exit from SS.Disabled.Error

- A peripheral upstream port shall transition to Rx.Detect upon PowerOn reset.
- A self-powered peripheral upstream port shall transition to SS.Disabled.Default upon detection of invalid VBUS.
- A peripheral upstream port shall remain in SS.Disabled.Error upon detection of USB 2.0 bus reset.

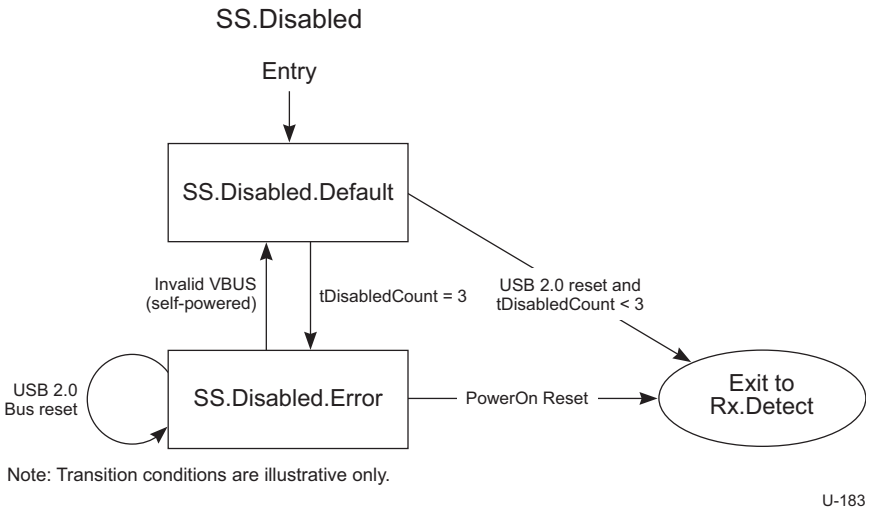


Figure 7-14. SS.Disabled Substate Machine

7.5.2 SS.Inactive

SS.Inactive is a state where a link has failed SuperSpeed operation. A downstream port can only exit from this state when directed, or upon detection of an absence of a far-end receiver termination (R_{RX-DC}) specified in Table 6-13, or upon a Warm Reset. An upstream port can only exit to Rx.Detect upon a Warm Reset, or upon detecting an absence of a far-end receiver termination (R_{RX-DC}) specified in Table 6-13.

During SS.Inactive, a port periodically performs a far-end receiver termination detection. If a disconnection is detected, a port will return to Rx.Detect. If a disconnect is not detected, the link will stay in SS.Inactive until software intervention.

7.5.2.1 SS.Inactive Substate Machines

SS.Inactive contains the following substate machines shown in Figure 7-15:

- SS.Inactive.Disconnect.Detect
- SS.Inactive.Quiet

7.5.2.2 SS.Inactive Requirements

- VBUS shall be present.
- The receiver termination shall meet the requirement (R_{RX-DC}) specified in Table 6-13.
- The transmitter common mode is not required to be within specification during this state.

7.5.2.3 SS.Inactive.Quiet

SS.Inactive.Quiet is a substate defined in which a port has disabled its far-end receiver termination detection so that extra power can be saved while waiting for software intervention.

7.5.2.3.1 SS.Inactive.Quiet Requirements

- The function of the far-end receiver termination detection shall be disabled.
- A 12-ms timer (tSSInactiveQuietTimeout) shall be started upon entry to the substate.

7.5.2.3.2 Exit from SS.Inactive.Quiet

- The port shall transition to SS.Inactive.Disconnect.Detect upon the 12-ms timer timeout (tSSInactiveQuietTimeout).
- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect when Warm Reset is issued.
- An upstream port shall transition to Rx.Detect upon detection of Warm Reset.

7.5.2.4 SS.Inactive.Disconnect.Detect

SS.Inactive.Disconnect.Detect is a substate in which a port will perform the far-end receiver termination detection in order to determine if its link partner is disconnected during SS.Inactive, or if the transition to SS.Inactive is due to a disconnect from its link partner.

7.5.2.4.1 SS.Inactive.Disconnect.Detect Requirements

The transmitter shall perform the far-end receiver termination detection described in Section 6.11.

7.5.2.4.2 Exit from SS.Inactive.Disconnect.Detect

- The port shall transition to Rx.Detect when a far-end low-impedance receiver termination (R_{RX-DC}) meeting specification defined in Table 6-13 is not detected.
- The port shall transition to SS.Inactive.Quiet when a far-end low-impedance receiver termination (R_{RX-DC}) meeting specification defined in Table 6-13 is detected.

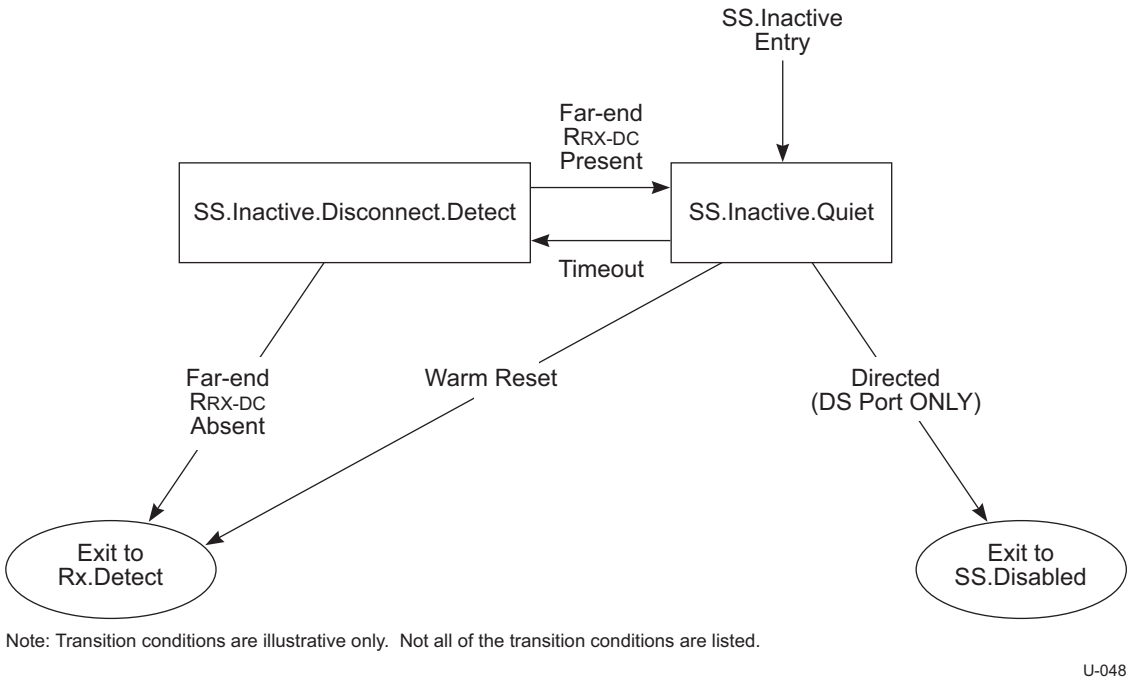


Figure 7-15. SS.Inactive Substate Machine

7.5.3 Rx.Detect

Rx.Detect is the power on state of the LTSSM for both a downstream port and an upstream port. It is also the state for a downstream port upon issuing a Warm Reset, and the state for an upstream port upon detecting a Warm Reset from any other link state except SS.Disabled. The purpose of Rx.Detect is to detect the impedance of far-end receiver termination to ground. Rx.Detect.Reset is a default reset state used by the two ports to synchronize the operation after a Warm Reset; this substate exits immediately if Warm Reset is not present. Rx.Detect.Active is a substate for far-end receiver termination detection. Rx.Detect.Quiet is a power saving substate in which the function of a far-end receiver termination detection is disabled. A port will perform the far-end receiver termination detection periodically during Rx.Detect.

7.5.3.1 Rx.Detect Substate Machines

Rx.Detect contains a substate machine shown in Figure 7-16 with the following substates:

- Rx.Detect.Reset
- Rx.Detect.Active
- Rx.Detect.Quiet

7.5.3.2 Rx.Detect Requirements

- The transmitter common mode is not required to be within specification during this state.
- The low-impedance receiver termination (R_{RX-DC}) defined in Table 6-13 shall be maintained.

7.5.3.3 Rx.Detect.Reset

Rx.Detect.Reset is a substate designed for the two ports to synchronize their operations on Warm Reset. In this substate, a downstream port shall generate Warm Reset when directed. If an upstream port enters Rx.Detect upon detection of Warm Reset, it shall remain in this substate until the completion of Warm Reset.

For a port entering Rx.Detect not due to a Warm Reset, it shall exit immediately.

7.5.3.3.1 Rx.Detect.Reset Requirements

If a port enters Rx.Detect upon a Warm Reset, the following requirements shall be applied. Refer to Section 6.9.3 for details.

- A downstream port shall transmit Warm Reset for the duration of tReset as defined in Table 6-21.

  Note: This includes the case when Hot Reset attempt fails in Recovery. Refer to Section 7.4.2 for details.
- An upstream port shall remain in this state until it detects the completion of Warm Reset.

7.5.3.3.2 Exit from Rx.Detect.Reset

- The port shall transition directly to Rx.Detect.Active if the entry to Rx.Detect is not due to a Warm Reset.

  Note: Warm Reset is not present during power-on.
- A downstream port shall transition to Rx.Detect.Active after it transmits Warm Reset for the duration of tReset as defined in Table 6-21.
- A downstream port shall transition to SS.Disabled when directed.
- An upstream port shall transition to Rx.Detect.Active when it receives no more LFPS Warm Reset signaling from the downstream port as defined in Section 6.9.3.

7.5.3.4 Rx.Detect.Active

Rx.Detect.Active is a substate to detect the presence of a SuperSpeed link partner. A port will perform a far-end receiver termination detection as defined in Section 6.11.

7.5.3.5 Rx.Detect.Active Requirements

- The transmitter shall initiate a far-end receiver termination detection described in Section 6.11.
- The number of far-end receiver termination detection events shall be counted by an upstream port. The detection of far-end receiver termination is defined in Section 6.11.

  Note: This count value is used by a peripheral device to determine when it needs to transition to SS.Disabled. It is also used by a hub to control its downstream port state machine. Refer to Section 10.3.1.1 for details.

7.5.3.6 Exit from Rx.Detect.Active

- The port shall transition to Polling upon detection of a far-end low-impedance receiver termination (R_{RX-DC}) defined in Table 6-13.
- A downstream port shall transition to Rx.Detect.Quiet when a far-end low-impedance receiver termination (R_{RX-DC}) defined in Table 6-13 is not detected.
- A downstream port shall transition to SS.Disabled when directed.
- An upstream port of a hub shall transition to Rx.Detect.Quiet when a far-end low-impedance receiver termination (R_{RX-DC}) defined in Table 6-13 is not detected.
- An upstream port of a peripheral device shall transition to Rx.Detect.Quiet when the following two conditions are met:
  1. A far-end low-impedance receiver termination (R_{RX-DC}) defined in Table 6-13 is not detected.
  2. The number of far-end receiver termination detection events is less than eight.
- An upstream port of a peripheral device shall transition to SS.Disabled when the following two conditions are met:
  1. A far-end low-impedance receiver termination (R_{RX-DC}) defined in Table 6-13 is not detected.
  2. The number of far-end receiver termination detection events has reached eight.

  Note: This limit on the number of the far-end receiver termination detections is to allow a SuperSpeed peripheral device on a legacy platform to transition to USB 2.0 after 80 ms.

7.5.3.7 Rx.Detect.Quiet

Rx.Detect.Quiet is a substate where a port has disabled its far-end receiver termination detection.

7.5.3.7.1 Rx.Detect.Quiet Requirements

- The far-end receiver termination detection shall be disabled.
- A 12-ms timer (tRxDetectQuietTimeout) shall be started upon entry to the substate.

7.5.3.7.2 Exit from Rx.Detect.Quiet

- The port shall transition to Rx.Detect.Active upon the 12-ms timer timeout (tRxDetectQuietTimeout).
- A downstream port shall transition to SS.Disabled when directed.

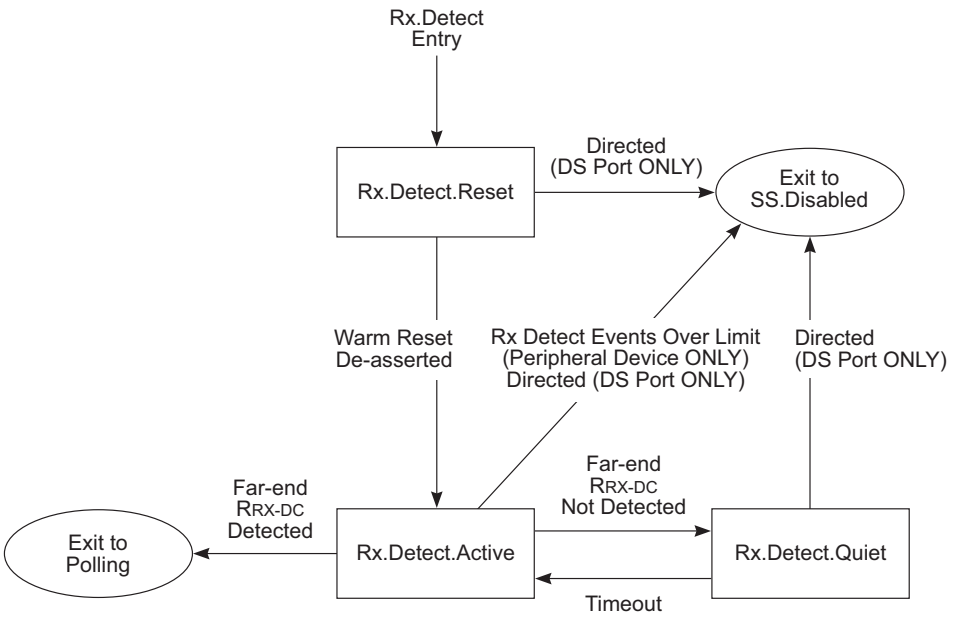


Figure 7-16. Rx.Detect Substate Machine

7.5.4 Polling

Polling is a state for link training. During Polling, a Polling.LFPS handshake shall take place between the two ports before the SuperSpeed training is started. Bit lock, symbol lock, and Rx equalization trainings are achieved using TSEQ, TS1, and TS2 training ordered sets.

7.5.4.1 Polling Substate Machines

Polling contains a substate machine shown in Figure 7-17 with the following substates:

- Polling.LFPS
- Polling.RxEQ
- Polling.Active
- Polling.Configuration
- Polling.Idle

7.5.4.2 Polling Requirements

The port shall maintain its low-impedance receiver termination (R_{RX-DC}) defined in Table 6-13.

7.5.4.3 Polling.LFPS

Polling.LFPS is a substate designed to establish the PHY's DC operating point, and to synchronize the operations between the two link partners after exiting from Rx.Detect.

7.5.4.3.1 Polling.LFPS Requirements

- Upon entry, a LFPS receiver shall be enabled to receive the Polling.LFPS signals defined in Section 6.9.1.
- Upon entry, a port shall establish its LFPS operating condition within 80 µs.
- A 360-ms timer (tPollingLFPSTimeout) shall be started upon entry to the substate.
- The operating condition of a SuperSpeed PHY shall be established when a port is ready to exit to Polling.RxEQ.
- A SuperSpeed receiver may optionally be enabled to receive TSEQ ordered sets for receiver equalizer training.

  Note: The port first entering Polling.RxEQ will start transmitting TSEQ ordered sets while the other port is still in Polling.LFPS. Enabling a SuperSpeed receiver in Polling.LFPS will allow a port to start the receiver equalizer training while completing the requirement for Polling.LFPS exit handshake.

7.5.4.3.2 Exit from Polling.LFPS

- The port shall transition to Polling.RxEQ when the following three conditions are met:
  1. At least 16 consecutive Polling.LFPS bursts meeting the Polling.LFPS specification defined in Section 6.9 are sent.
  2. Two consecutive Polling.LFPS bursts are received.
  3. Four consecutive Polling.LFPS bursts are sent after receiving one Polling.LFPS burst.
- The port shall transition to Compliance Mode upon the 360-ms timer timeout (tPollingLFPSTimeout) and the following two conditions are met:
  1. The port has never successfully completed Polling.LFPS after PowerOn Reset.
  2. The condition to transition to Polling.RxEQ is not met.

  Note: If the very first attempt in Polling.LFPS handshake fails after PowerOn Reset, it implies that a passive test load may be present and compliance test should be initiated. If the very first attempt in Polling.LFPS handshake succeeds after PowerOn Reset, it implies the presence of the SuperSpeed ports on each side of the link and no compliance test is intended. Therefore, any subsequent handshake timeout in Polling.LFPS when the link is retrained is only an indication of link training failure, not a signal to enter Compliance Mode.
- A downstream port shall transition to Rx.Detect upon the 360-ms timer timeout (tPollingLFPSTimeout) after having trained once since PowerOn Reset and the conditions to transition to Polling.RxEQ are not met.
- An upstream port of a hub shall transition to Rx.Detect upon the 360-ms timeout (tPollingLFPSTimeout) after having trained once since PowerOn Reset and the conditions to transition to Polling.RxEQ are not met.
- A peripheral device shall transition to SS.Disabled upon the 360-ms timeout (tPollingLFPSTimeout) after having trained once since PowerOn Reset and the conditions to transition to Polling.RxEQ are not met.
- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.

7.5.4.4 Polling.RxEQ

Polling.RxEQ is a substate for receiver equalization training. A port is required to complete its receiver equalization training.

7.5.4.4.1 Polling.RxEQ Requirements

- The detection and correction of the lane polarity inversion shall be enabled, as is described in Section 6.4.2.
- The port shall transmit the TSEQ ordered sets defined in Table 6-2.
- The port shall complete receiver equalizer training upon exit from this substate.

Note: A situation may exist where the port entering Polling.RxEQ earlier is transmitting TSEQ ordered sets while its link partner is still sending Polling.LFPS to satisfy the exit conditions from Polling.LFPS to Polling.RxEQ. In this situation, if its link partner is in electrical idle, near-end cross talk may cause the port to train its Rx equalizer using its own TSEQ ordered sets. To avoid a receiver from training itself, a port may either ignore the beginning part (about 30 µs) of the TSEQ ordered sets, or continue the equalizer training until it completes the transmission of TSEQ ordered sets.

7.5.4.4.2 Exit from Polling.RxEQ

- The port shall transition to Polling.Active after 65,536 consecutive TSEQ ordered sets defined in Table 6-2 are transmitted.
- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.

7.5.4.5 Polling.Active

Polling.Active is a substate that continues the link's SuperSpeed training.

7.5.4.5.1 Polling.Active Requirements

- A 12-ms timer (tPollingActiveTimeout) shall be started upon entry to this substate.
- The port shall transmit TS1 ordered sets.
- The receiver is in training using TS1 or TS2 ordered sets.

Note: Depending on the link condition and different receiver implementations, one port's receiver may train faster than the other. When this occurs, the port whose receiver trains first will enter Polling.Configuration and start transmitting TS2 ordered sets while the port whose receiver is not yet trained is still in Polling.Active using TS2 ordered sets to train its receiver.

7.5.4.5.2 Exit from Polling.Active

- The port shall transition to Polling.Configuration upon receiving eight consecutive and identical TS1 or TS2 ordered sets.
- A downstream port shall transition to Rx.Detect upon the 12-ms timer timeout (tPollingActiveTimeout) and the conditions to transition to Polling.Configuration are not met.
- An upstream port of a hub shall transition to Rx.Detect upon the 12-ms timer timeout (tPollingActiveTimeout) and the conditions to transition to Polling.Configuration are not met.
- An upstream port of a peripheral device shall transition to SS.Disabled upon the 12-ms timer timeout (tPollingActiveTimeout) and the conditions to transition to Polling.Configuration are not met.
- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.

7.5.4.6 Polling.Configuration

Polling.Configuration is a substate where the two link partners complete the SuperSpeed training.

7.5.4.6.1 Polling.Configuration Requirements

- The port shall transmit identical TS2 ordered sets upon entry to this substate and set the link configuration field in the TS2 ordered set based on the following.
  1. When directed, a downstream port shall set Reset bit in the TS2 ordered set.
     Note: An upstream port can only set the Reset bit in the TS2 ordered set when in Hot Reset. Active. Refer to Section 7.5.12.3 for details.
  2. When directed, the port shall set Loopback bit in the TS2 ordered set.
  3. When directed, the port shall set the Disabling Scrambling bit in the TS2 ordered set.
- A 12-ms timer (tPollingConfigurationTimeout) shall be started upon entry to this substate.

7.5.4.6.2 Exit from Polling.Configuration

- The port shall transition to Polling.Idle when the following two conditions are met:
  1. Eight consecutive and identical TS2 ordered sets are received.
  2. Sixteen TS2 ordered sets are sent after receiving the first of the eight consecutive and identical TS2 ordered sets.
- A downstream port shall transition to Rx.Detect upon the 12-ms timer timeout (tPollingConfigurationTimeout) and the conditions to transition to Polling.Idle are not met.
- An upstream port of a hub shall transition to Rx.Detect upon the 12-ms timer timeout (tPollingConfigurationTimeout) and the conditions to transition to Polling.Idle are not met.

- An upstream port of a peripheral device shall transition to SS.Disabled upon the 12-ms timer timeout (tPollingConfigurationTimeout) and the conditions to transition to Polling.Idle are not met.
- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.

7.5.4.7 Polling.Idle

Polling.Idle is a substate where the port decodes the TS2 ordered set received in Polling.Configuration and determines the next state.

7.5.4.7.1 Polling.Idle Requirements

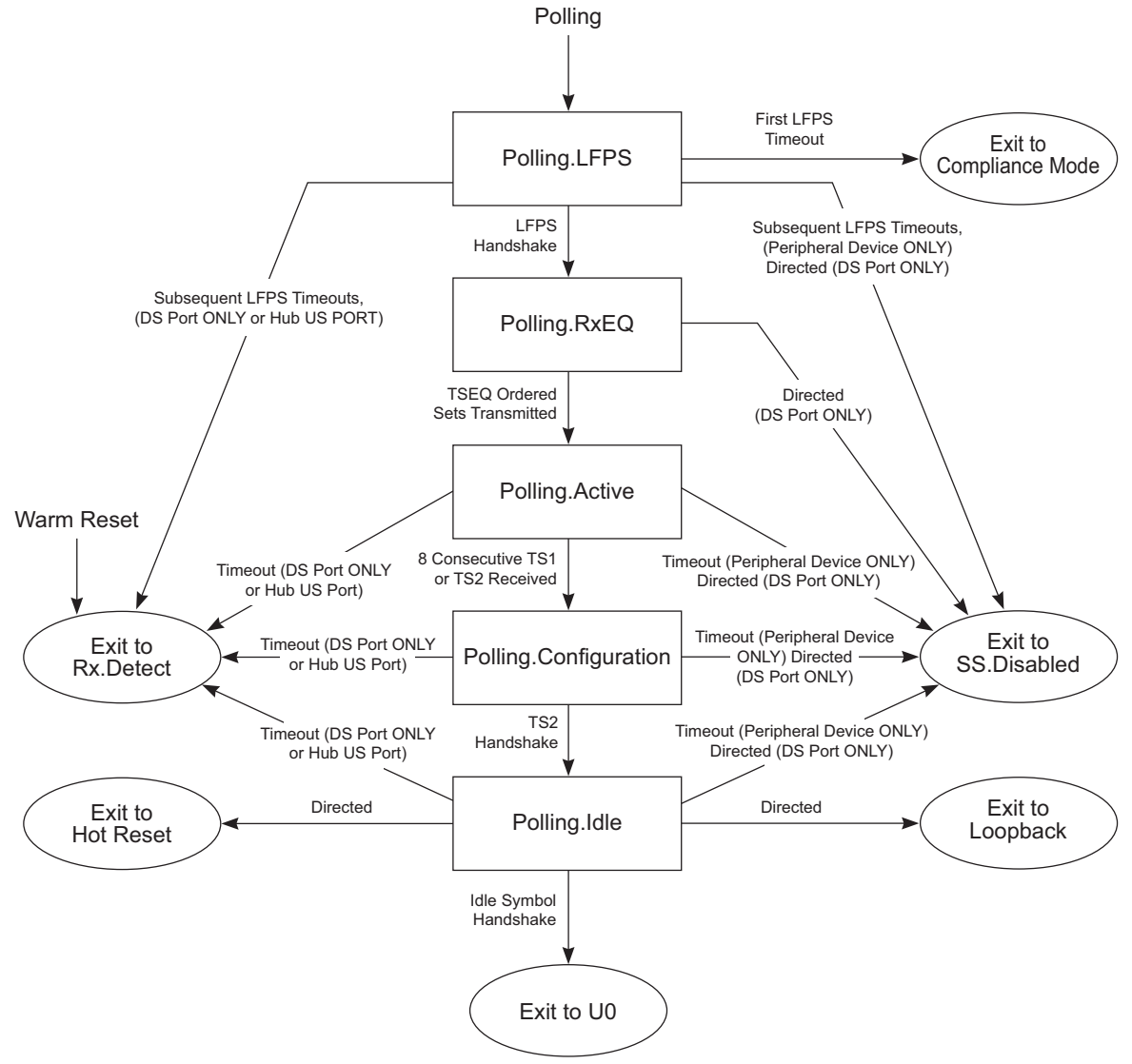
- The port shall decode the TS2 ordered set received during Polling.Configuration and proceeds to the next state.
- A downstream port shall reset its Link Error Count.
- An upstream port shall reset its port configuration information to default values. Refer to Sections 8.4.5 and 8.4.6 for details.
- The port shall enable the scrambling by default if the Disabling Scrambling bit is not asserted in the TS2 ordered set received in Polling.configuration.
- The port shall disable the scrambling if directed, or if the Disabling Scrambling bit is asserted in the TS2 ordered set received in Polling.configuration.
- The port shall transmit Idle Symbols if the next state is U0. The port may transmit Idle Symbols if the next state is Loopback or HotReset.
- A 2-ms timer (tPollingIdleTimeout) shall be started upon entry to this state.
- The port shall be able to receive the Header Sequence Number Advertisement from its link partner.

  Note: The exit time difference between the two ports will result in one port entering U0 first and starting the Header Sequence Number Advertisement while the other port is still in Polling.Idle.

7.5.4.7.2 Exit from Polling.Idle

- The port shall transition to Loopback when directed as a loopback master and the port is capable of being a loopback master.
- The port shall transition to Loopback as a loopback slave if the Loopback bit is asserted in the TS2 ordered set received in Polling.Configuration.
- A downstream port shall transition to Hot Reset when directed.
- An upstream port shall transition to Hot Reset when the Reset bit is asserted in the TS2 ordered set received in Polling.Configuration.
- The port shall transition to U0 when the following two conditions are met:
  1. Eight consecutive Idle Symbols are received.
  2. Sixteen Idle Symbols are sent after receiving one Idle Symbol.
- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect upon the 2-ms timer timeout (tPollingIdleTimeout) and the conditions to transition to U0 are not met.

- An upstream port of a hub shall transition to Rx.Detect upon the 2-ms timer timeout (tPollingIdleTimeout) and the conditions to transition to U0 are not met.
- An upstream port of a peripheral device shall transition to SS.Disabled upon the 2-ms timer timeout (tPollingIdleTimeout) and the conditions to transition to U0 are not met.
- A downstream port shall transition to Rx.Detect when Warm Reset is directed.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.



Note: Transition conditions are illustrative only. Not all of the transition conditions are listed.

U-050

Figure 7-17. Polling Substate Machine

7.5.5 Compliance Mode

Compliance Mode is used to test the transmitter for compliance to voltage and timing specifications. Several different test patterns are transmitted as defined in Table 6-7. Compliance Mode does not contain any substate machines.

7.5.5.1 Compliance Mode Requirements

- The port shall maintain its low-impedance receiver termination (R_{RX-DC}) defined in Table 6-13.
- The LFPS receiver is used to control the test pattern sequencing.
- Upon entry to Compliance Mode, the port shall wait until its SuperSpeed Tx DC common mode voltage meets the $V_{TX-DC-CM}$ specification defined in Table 6-11 before it starts to send the first compliance test pattern defined in Table 6-7.
- The port shall transmit the next compliance test pattern continuously upon detection of a Ping.LFPS as defined in Section 6.9.1.
- The port shall transmit the first compliance test pattern continuously upon detection of a Ping.LFPS and the test pattern has reached the final test pattern.

7.5.5.2 Exit from Compliance Mode

- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect upon detection of Warm Reset.
- A downstream port shall transition to SS.Disabled when directed.

7.5.6 U0

U0 is the normal operational state where packets can be transmitted and received. U0 does not contain any substate machines.

7.5.6.1 U0 Requirements

- The port shall meet the transmitter specifications defined in Table 6-10.
- The port shall maintain the low-impedance receiver termination (R_{RX-DC}) defined in Table 6-13.
- The LFPS receiver shall be enabled.
- The port shall enable a 1-ms timer (tU0RecoveryTimeout) to measure the time interval between two consecutive link commands. This timer will be reset and restarted every time a link command is received.
- The port shall enable a 10-µs timer (tU0LTimeout). This timer shall be reset when the first symbol of any link command or packet is sent and restarted after the last symbol of any link command or packet is sent. This timer shall be active when the link is in logical idle.
- A downstream port shall transmit a single LDN when the 10-µs timer (tU0LTimeout) expires.
- An upstream port shall transmit a single LUP when the 10-µs timer (tU0LTimeout) expires.

7.5.6.2 Exit from U0

- The port shall transition to U1 upon successful completion of LGO_U1 entry sequence. Refer to Section 7.2.4.2 for details.
- The port shall transition to U2 upon successful completion of LGO_U2 entry sequence. Refer to Section 7.2.4.2 for details.
- The port shall transition to U3 upon successful completion of LGO_U3 entry sequence. Refer to Section 7.2.4.2 for details.

- A downstream port shall transition to SS.Inactive when it fails U3 entry on three consecutive attempts.
- The port shall transition to Recovery upon any errors stated in Section 7.3 that will cause a link to transition to Recovery.
- The port shall transition to Recovery upon detection of a TS1 ordered set.
- The port shall transition to Recovery when directed.
- The port shall transition to SS.Inactive when PENDING_HP_TIMER times out for the fourth consecutive time.

  Note: This implies the link has transitioned to Recovery for three consecutive times and each time the transition to Recovery is due to PENDING_HP_TIMER timeout.
- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to SS.Inactive when directed.
- An upstream port shall transition to SS.Disabled when directed.

  Note: After entry to U0 and the successful completion of training and link initialization, both ports are required to exchange port capabilities information using Port Capability LMPs within tPortConfiguration time as defined in Section 8.4.5. This includes the following scenarios:
  1. Entry to U0 from polling directly;
  2. Entry to U0 indirectly from Polling through Hot Reset;
  3. Entry to U0 from Recovery and port configuration has not been successfully completed after exiting from Polling. In this case, both ports shall continue the port configuration process by completing the remaining LMP exchanges.

  If the port has not received a Port Capability LMP within tPortConfiguration time, a downstream port shall be directed to transition to SS.Inactive and an upstream port shall be directed to transition to SS.Disabled.
- A downstream port shall transition to Recovery upon not receiving any link commands within 1 ms (tU0RecoveryTimeout).

  Note: Not receiving any link commands including LUP within 1 ms implies either a link is under serious error condition, or an upstream port has been removed. To accommodate for both situations, a downstream port will transition to Recovery and attempt to retrain the link. If the retraining fails, it will then transition to SS.Inactive. During SS.Inactive, a downstream port will attempt a far-end receiver termination detection. If it determines that a far-end low-impedance receiver termination (R_{RX-DC}) defined in Table 6-13 is not present, it will enter Rx.Detect. Otherwise, it will wait for software intervention.
- An upstream port shall transition to Recovery if it does not receive any link command or any packet (as specified in Section 7.2.4.1.4) within 1 ms (tU0RecoveryTimeout).
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.
- An upstream port shall transition to SS.Disabled upon detection of V_{BUS} off.

  Note: this condition only applies to a self-powered upstream port. SS.Disabled is a logical power-off state for a self-powered upstream port.

7.5.7 U1

U1 is a low power state where no packets are to be transmitted and both ports agree to enter a link state where a SuperSpeed PHY can be placed into a low power state.

U1 does not contain any substates. Transitions to other states are shown in Figure 7-18.

7.5.7.1 U1 Requirements

- The SuperSpeed transmitter DC common mode voltage shall be within specification ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) defined in Table 6-11.
- The port shall maintain its low-impedance receiver termination ($R_{RX\text{-}DC}$) defined in Table 6-13.
- The port shall enable its U1 exit detect functionality as defined in Section 6.9.2.
- The port shall enable its LFPS transmitter when it initiates the exit from U1.
- The port shall enable its U2 inactivity timer upon entry to this state if the U2 inactivity timer has a non-zero timeout value.
- A downstream port shall enable its Ping.LFPS detection.
- A downstream port shall enable a 300-ms timer (tU1PingTimeout). This timer will be reset and restarted when a Ping.LFPS is received.
- An upstream port shall transmit Ping.LFPS as defined in Table 6-20.

7.5.7.2 Exit from U1

- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect when the 300-ms timer (tU1PingTimeout) expires.
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.
- A self-powered upstream port shall transition to SS.Disabled upon not detecting valid Vbus as defined in Section 11.4.5.
- The port shall transition to U2 upon the timeout of the U2 inactivity timer defined in Sections 10.4.2.4 and 10.6.2.4.
- The port shall transition to Recovery upon successful completion of a LFPS handshake meeting the U1 LFPS exit handshake signaling in Section 6.9.2.
- The port shall transition to SS.Inactive upon the 2-ms (tNoLFPSResponseTimeout) LFPS handshake timer timeout and a successful LFPS handshake meeting the U1 LFPS exit handshake signaling in Section 6.9.2 is not achieved.

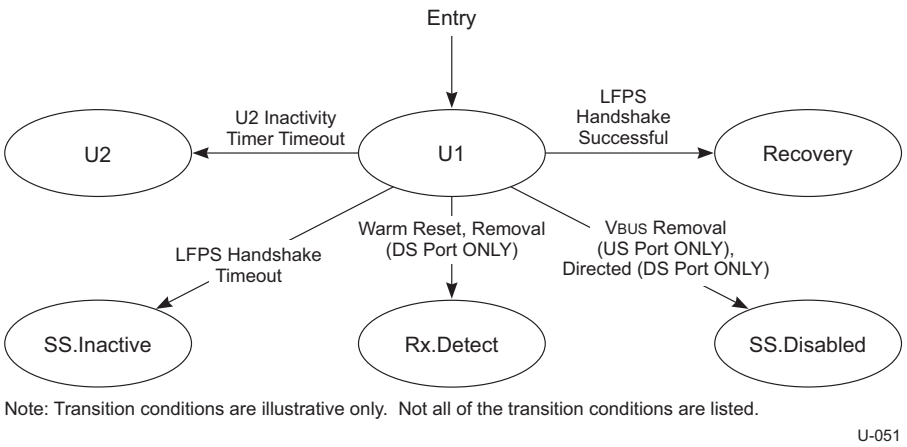


Figure 7-18. U1

7.5.8 U2

U2 is a link state where more power saving opportunities are allowed compare to U1, but with an increased exit latency.

U2 does not contain any substates. The transitions to other states are shown in Figure 7-19.

7.5.8.1 U2 Requirements

- The SuperSpeed transmitter DC common mode voltage does not need to be within specification ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) defined in Table 6-10.
- The port shall maintain its low-impedance receiver termination ($R_{RX\text{-}DC}$) defined in Table 6-13.
- When a downstream port is in U2, its upstream port may be in U1 or U2. If the upstream port is in U1, it will send Ping.LFPS periodically. A downstream port shall differentiate between Ping.LFPS and U1 LFPS exit handshake signaling.
- The port shall enable its U2 exit detect functionality as defined in Section 6.9.2.
- The port shall enable its LFPS transmitter when it initiates the exit from U2.
- A downstream port shall perform a far-end receiver termination detection every 100 ms (tU2RxdetDelay).

7.5.8.2 Exit from U2

- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect upon detection of a far-end high-impedance receiver termination ($Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}POS}$) defined in Table 6-13.
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.
- A self-powered upstream port shall transition to SS.Disabled upon not detecting valid Vbus as defined in Section 11.4.5.
- The port shall transition to Recovery upon successful completion of a LFPS handshake meeting the U2 LFPS exit signaling defined in Section 6.9.2.

- The port shall transition to SS.Inactive upon the 2-ms LFPS handshake timer timeout (tNoLFPSResponseTimeout) and a successful LFPS handshake meeting the U2 LFPS exit handshake signaling in Section 6.9.2 is not achieved.

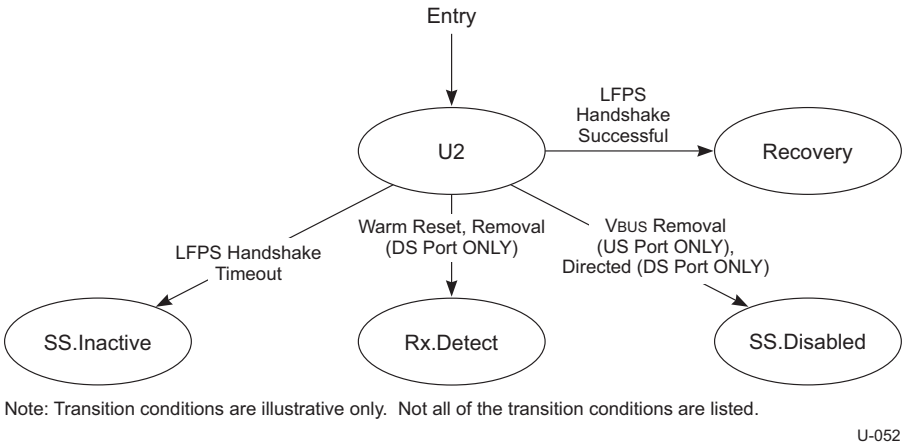


Figure 7-19. U2

7.5.9 U3

U3 is a link state where a device is put into a suspend state. Significant link and device powers are saved.

U3 does not contain any substates. Transitions to other states are shown in Figure 7-20.

7.5.9.1 U3 Requirements

- The SuperSpeed transmitter DC common mode voltage does not need to be within specification ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) defined in Table 6-10.
- The port shall maintain its low-impedance receiver termination ($R_{RX\text{-}DC}$) defined in Table 6-13.
- LFPS Ping detection shall be disabled.
- The port shall enable its U3 wakeup detect functionality as defined in Section 6.9.2.
- The port shall enable its LFPS transmitter when it initiates the exit from U3.
- A downstream port shall perform a far-end receiver termination detection every 100 ms (tU3RxdetDelay).
- The port not able to respond to U3 LFPS wakeup within tNoLFPSResponseTimeout may initiate U3 LFPS wakeup when it is ready to return to U0.

7.5.9.2 Exit from U3

- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect upon detection of a far-end high-impedance receiver termination ($Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}POS}$) defined in Table 6-13.
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.

- A self-powered upstream port shall transition to SS.Disabled upon not detecting valid VBUS as defined in Section 11.4.5.
- The port shall transition to Recovery upon successful completion of a LFPS handshake meeting the U3 wakeup signaling defined in Section 6.9.2.
- The port shall remain in U3 when the 10-ms LFPS handshake timer times out (tNoLFPSResponseTimeout) and a successful LFPS handshake meeting the U3 wakeup handshake signaling in Section 6.9.2 is not achieved. 100 ms (tU3WakeupRetryDelay) after an unsuccessful LFPS handshake and the requirement to exit U3 still exists, then the port shall initiate the U3 wakeup LFPS Handshake signaling to wake up the host.

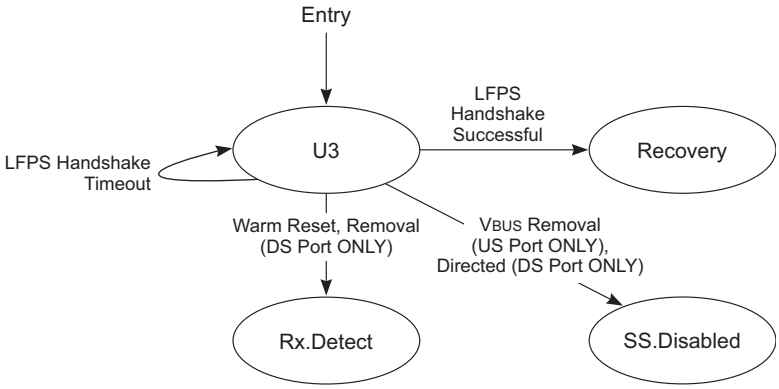


Figure 7-20. U3

7.5.10 Recovery

The Recovery link state is entered to retrain the link, or to perform Hot Reset, or to switch to Loopback mode. In order to retrain the link and also minimize the recovery latency, the two link partners do not train the receiver equalizers. Instead, the last trained equalizer configurations are maintained. Only TS1 and TS2 ordered sets are transmitted to synchronize the link and to exchange the link configuration information defined in Table 6-5.

7.5.10.1 Recovery Substate Machines

Recovery contains a substate machine shown in Figure 7-21 with the following substates:

- Recovery.Active
- Recovery.Configuration
- Recovery.Idle

7.5.10.2 Recovery Requirements

- The port shall meet the transmitter specifications as defined in Table 6-10.
- The port shall maintain the low-impedance receiver termination (R_{RX-DC}) as defined in Table 6-13.
- All header packets in the Tx Header Buffers and the Rx Header Buffers shall be handled based on the requirements specified in Section 7.2.4.

7.5.10.3 Recovery.Active

Recovery.Active is a substate to train the SuperSpeed link by transmitting the TS1 ordered sets.

7.5.10.3.1 Recovery.Active Requirements

- A 12-ms timer (tRecoveryActiveTimeout) shall be started upon entry to this substate.
- The port shall transmit the TS1 ordered sets upon entry to this substate.
- The port shall train its receiver with TS1 or TS2 ordered sets.

Note: Depending on the link condition and different receiver implementations, one port's receiver may train faster than the other. When this occurs, the port whose receiver trains first will enter Recovery.Configuration and start transmitting TS2 ordered sets while the port whose receiver is not yet trained is still in Recovery.Active using the TS2 ordered sets to train its receiver.

7.5.10.3.2 Exit from Recovery.Active

- The port shall transition to Recovery.Configuration after eight consecutive and identical TS1 or TS2 ordered sets are received.
- The port shall transition to SS.Inactive when the following conditions are met:
  1. Either the Ux_EXIT_TIMER or the 12-ms timer (tRecoveryActiveTimeout) times out.
  2. For a downstream port, the transition to Recovery is not to attempt a Hot Reset.
- A downstream port shall transition to Rx.Detect when the following conditions are met:
  1. Either the Ux_EXIT_TIMER or the 12-ms timer (tRecoveryActiveTimeout) times out.
  2. The transition to Recovery is to attempt a Hot Reset.
- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.

7.5.10.4 Recovery.Configuration

Recovery.Configuration is a substate designed to allow the two link partners to achieve the SuperSpeed handshake by exchanging the TS2 ordered sets.

7.5.10.4.1 Recovery.Configuration Requirements

- The port shall transmit identical TS2 ordered sets upon entry to this substate and set the link configuration field in the TS2 ordered set based on the following:
  1. When directed, a downstream port shall set Reset bit in the TS2 ordered set.
     Note: An upstream port can only set the Reset bit in the TS2 ordered set when in Hot Reset. Active. Refer to Section 7.5.12.3 for details.
  2. When directed, the port shall set Loopback bit in the TS2 ordered set.
  3. When directed, the port shall set the Disabling Scrambling bit in the TS2 ordered set.
- A 6-ms timer (tRecoveryConfigurationTimeout) shall be started upon entry to this substate.

7.5.10.4.2 Exit from Recovery.Configuration

- The port shall transition to Recovery.Idle after the following two conditions are met:
  1. Eight consecutive and identical TS2 ordered sets are received.
  2. Sixteen TS2 ordered sets are sent after receiving the first of the eight consecutive and identical TS2 ordered sets.
- The port shall transition to SS.Inactive when the following conditions are met:
  1. Either the Ux_EXIT_TIMER or the 6-ms timer (tRecoveryConfigurationTimeout) times out.
  2. For a downstream port, the transition to Recovery is not to attempt a Hot Reset.
- A downstream port shall transition to Rx.Detect when the following conditions are met:
  1. Either the Ux_EXIT_TIMER or the 6-ms timer (tRecoveryConfigurationTimeout) times out.
  2. The transition to Recovery is to attempt a Hot Reset.
- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.

7.5.10.5 Recovery.Idle

Recovery.Idle is a substate where a port decodes the link configuration field defined in the TS2 ordered set received during Recovery.Configuration and determines the next state.

7.5.10.5.1 Recovery.Idle Requirements

- A 2-ms timer (tRecoveryIdleTimeout) shall be started upon entry to this substate.
- The port shall transmit Idle Symbols if the next state is U0. The port may transmit Idle Symbols if the next state is Loopback or HotReset.
- The port shall decode the link configuration field defined in the TS2 ordered sets received during Recovery.Configuration and proceed to the next state.
- The port shall enable the scrambling by default if the Disabling Scrambling bit is not asserted in the TS2 ordered set received in Recovery.configuration.
- The port shall disable the scrambling if directed, or if the Disabling Scrambling bit is asserted in the TS2 ordered set received in Recovery.configuration.
- The port shall be able to receive the Header Sequence Number Advertisement from its link partner.

Note: The exit time difference between the two ports will result in one port entering U0 first and starting the Header Sequence Number Advertisement while the other port is still in Recovery.Idle.

7.5.10.5.2 Exit from Recovery.Idle

- The port shall transition to Loopback when directed as a loopback master and the port is capable of being a loopback master.
- The port shall transition to Loopback as a loopback slave if the Loopback bit is asserted in TS2 ordered sets.
- The port shall transition to U0 when the following two conditions are met:
  1. Eight consecutive Idle Symbols are received.
  2. Sixteen Idle Symbols are sent after receiving one Idle Symbol.
- The port shall transition to SS.Inactive when one of the following timers times out and the conditions to transition to U0 are not met:
  1. Ux_EXIT_TIMER
  2. The 2-ms timer (tRecoveryIdleTimeout)
- A downstream port shall transition to Hot Reset when directed.
- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.
- An upstream port shall transition to Hot Reset if the Reset bit is asserted in TS2 ordered sets.

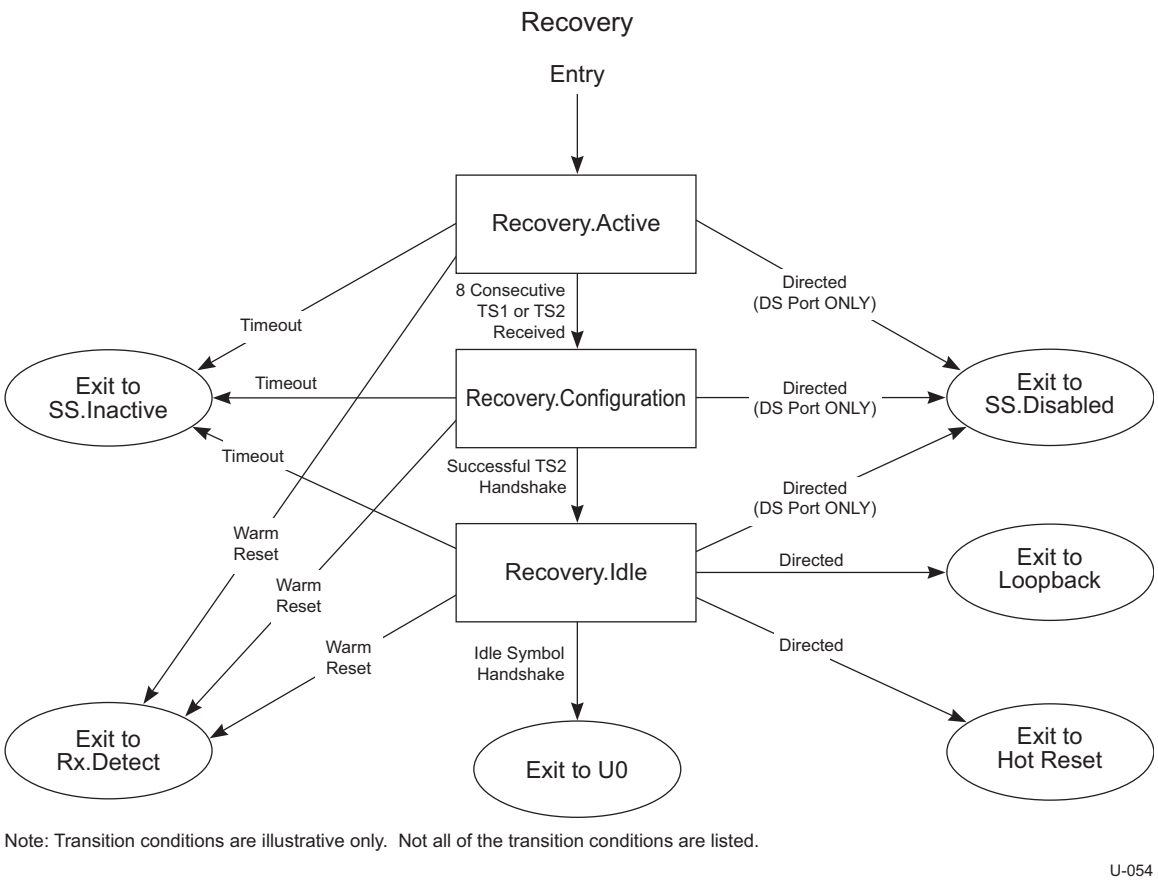


Figure 7-21. Recovery Substate Machine

7.5.11 Loopback

Loopback is intended for test and fault isolation. Loopback includes a bit error rate test (BERT) state machine, described in Chapter 6.

A loopback master is the port requesting loopback. A loopback slave is the port that retransmits the symbols received from the loopback master.

During Loopback.Active, the loopback slave must support the BERT protocol described in Chapter 6. The loopback slave must respond to the command for BERT error counter reset and BERT report error count. The loopback slave must check the incoming data for the loopback data pattern.

7.5.11.1 Loopback Substate Machines

Loopback contains a substate machine shown in Figure 7-22 with the following substates:

- Loopback.Active
- Loopback.Exit

7.5.11.2 Loopback Requirements

- There shall be one loopback master and one loopback slave. The loopback master is the port that has the Loopback bit asserted in TS2 ordered sets.
- The port shall maintain its transmitter specifications defined in Table 6-10.
- The port shall maintain its low-impedance receiver termination (R_{RX-DC}) defined in Table 6-13.

7.5.11.3 Loopback.Active

Loopback.Active is a substate where the loopback test is active. The loopback master is sending data/commands to its loopback slave. The loopback slave is either looping back the data or detecting/executing the commands it received from the loopback master.

7.5.11.3.1 Loopback.Active Requirements

- The loopback master shall send valid 8b/10b data with SKPs as necessary.
- The loopback slave shall retransmit the received 10-bit symbols.
- The loopback slave shall not modify the received 10-bit symbols, other than lane polarity inversion if necessary, and SKP ordered set, which may be added or dropped as required.

  Note: This implies that the loopback slave should disable or bypass its own 8b/10b encoder/decoder and scrambler/descrambler.
- The loopback slave must process the BERT commands as defined in Section 6.8.4.
- The LFPS receiver shall be enabled.

7.5.11.3.2 Exit from Loopback.Active

- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.
- When directed, the loopback master shall transition to Loopback.Exit.
- The loopback slave shall transition to Loopback.Exit upon detection of Loopback LFPS exit handshake signal meeting Loopback LFPS exit signaling defined in Section 6.9.2.

7.5.11.4 Loopback.Exit

Loopback.Exit is a substate where a loopback master has completed the loopback test and starts the exit from Loopback.

7.5.11.4.1 Loopback.Exit Requirements

- A 2-ms timer (tLoopbackExitTimeout) shall be started upon entry to the substate.
- The LFPS transmitter and the LFPS receiver shall be enabled.
- The port shall transmit and receive Loopback LFPS exit handshake signal defined in Section 6.9.2.

7.5.11.4.2 Exit from Loopback.Exit

- The port shall transition to Rx.Detect upon a successful Loopback LFPS exit handshake defined in Section 6.9.2.
- The port shall transition to SS.Inactive upon the 2-ms timer timeout (tLoopbackExitTimeout) and the condition to transition to Rx.Detect is not met.
- A downstream port shall transition to SS.Disabled when directed.

- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.

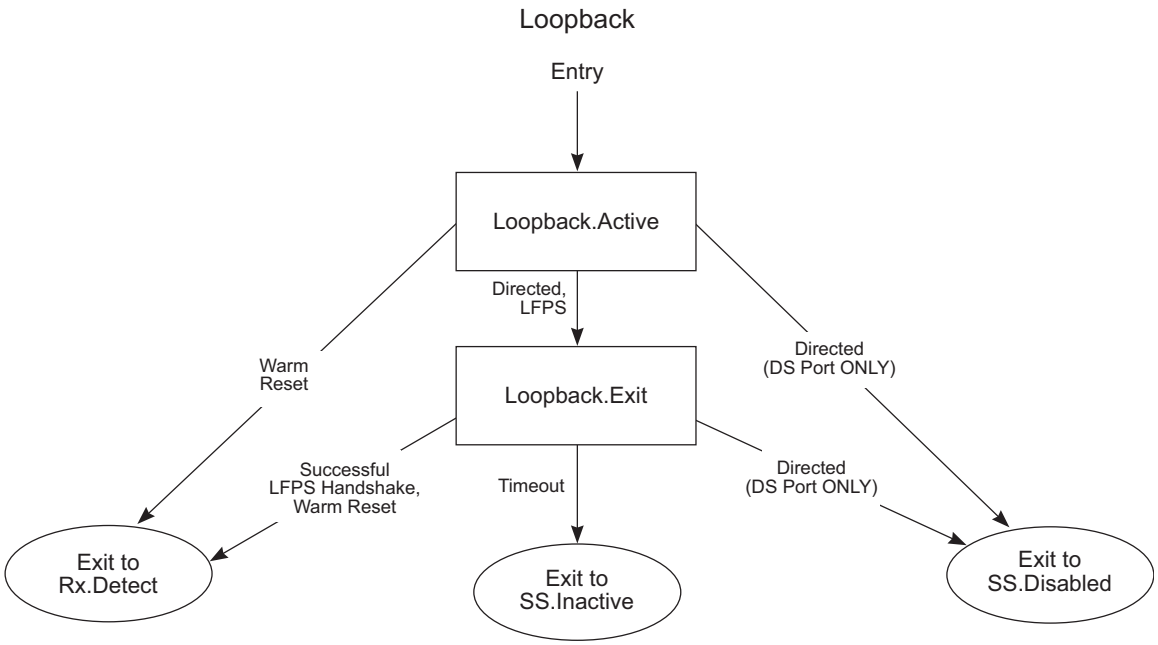


Figure 7-22. Loopback Substate Machine

7.5.12 Hot Reset

Only a downstream port can be directed to initiate a Hot Reset.

When the downstream port initiates reset, it shall transmit TS2 ordered sets with the Reset bit asserted. The upstream port shall respond by sending the TS2 ordered sets with Reset bit asserted. Upon completion of Hot Reset processing, the upstream port shall signal the downstream port by sending the TS2 ordered sets with the Reset bit de-asserted. The downstream port shall respond with the Reset bit de-asserted in the TS2 ordered sets. Once both ports receive the TS2 ordered sets with the Reset bit de-asserted, they shall exit from Hot Reset.Active and transition to Hot Reset.Exit. Once a successful idle symbol handshake is achieved, the port shall return to U0.

7.5.12.1 Hot Reset Substate Machines

Hot Reset contains a substate machine shown in Figure 7-23 with the following substates:

- Hot Reset Active
- Hot Reset.Exit

7.5.12.2 Hot Reset Requirements

- A downstream port shall reset its Link Error Count as defined in Section 7.4.2.
- A downstream port shall reset its PM timers and the associated U1 and U2 timeout values to zero.
- The port Configuration information shall remain unchanged (refer to Section 8.4.6 for details).
- The port shall maintain its transmitter specifications defined in Table 6-10.
- The port shall maintain its low-impedance receiver termination (R_{RX-DC}) defined in Table 6-13.

7.5.12.3 Hot Reset.Active

Hot Reset.Active is a substate where a port will perform the reset as defined in Section 7.4.12.2.

7.5.12.3.1 Hot Reset.Active Requirements

- Upon entry to this substate, the port shall first transmit at least 16 TS2 ordered sets continuously with the Reset bit asserted.

  Note: Depending on the time delay between the two ports entering Hot Reset, when the downstream port is transmitting the first 16 TS2 ordered sets with the Reset bit asserted, it may still receive part of the TS2 ordered sets from the upstream port exiting from Polling.Configuration or Recovery.Configuration. The downstream port shall ignore those TS2 ordered sets. Also upon entry to this substate, both ports shall ignore the Disabling Scrambling bit in the link configuration field of the TS2 Ordered Set. This bit is only decoded in Polling.Idle or Recovery.Idle.
- A 12-ms timer (tHotResetActiveTimeout) shall be started upon entry to this substate.
- A downstream port shall continue to transmit TS2 ordered sets with the Reset bit asserted until the upstream port transitions from sending TS2 ordered sets with the Reset bit asserted to sending the TS2 ordered sets with the Reset bit de-asserted.
- An upstream port shall transmit TS2 ordered sets with the Reset bit asserted while performing the Hot Reset.

- An upstream port shall transmit TS2 ordered sets with the Reset bit de-asserted after completing the Hot Reset.
- The port shall perform Hot Reset described in Hot Reset requirement of this section.

7.5.12.3.2 Exit from Hot Reset.Active

- The port shall transition to Hot Reset.Exit when the following three conditions are met.
  1. At least 16 TS2 ordered sets with the Reset bit asserted are transmitted.
  2. Two consecutive TS2 ordered sets are received with the Reset bit de-asserted.
  3. Four consecutive TS2 ordered sets with the Reset bit de-asserted are sent after receiving one TS2 ordered set with the Reset bit de-asserted.
- The port shall transition to SS.Inactive upon the 12-ms timer timeout (tHotResetActiveTimeout) and the conditions to transition to Hot Reset.Exit are not met.
- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.

7.5.12.4 Hot Reset.Exit

Hot Reset.Exit is a substate where the port has completed Hot Reset and is ready to exit from Hot Reset.

7.5.12.4.1 Hot Reset.Exit Requirements

- The port shall transmit idle symbols.
- A 2-ms timer (tHotResetExitTimeout) shall be started upon entry to this substate.
- The port shall be able to receive the Header Sequence Number Advertisement from its link partner.

Note: The exit time difference between the two ports will result in one port entering U0 first and starting the Header Sequence Number Advertisement while the other port is still in Hot Reset.Idle.

7.5.12.4.2 Exit from Hot Reset.Exit

- The port shall transition to U0 when the following two conditions are met:
  1. Eight consecutive Idle Symbols are received.
  2. Sixteen Idle Symbols are sent after receiving one Idle Symbol.
- The port shall transition to SS.Inactive upon the 2-ms timer timeout (tHotResetExitTimeout) and the conditions to transition to U0 are not met.
- A downstream port shall transition to SS.Disabled when directed.
- A downstream port shall transition to Rx.Detect when directed to issue Warm Reset.
- An upstream port shall transition to Rx.Detect when Warm Reset is detected.

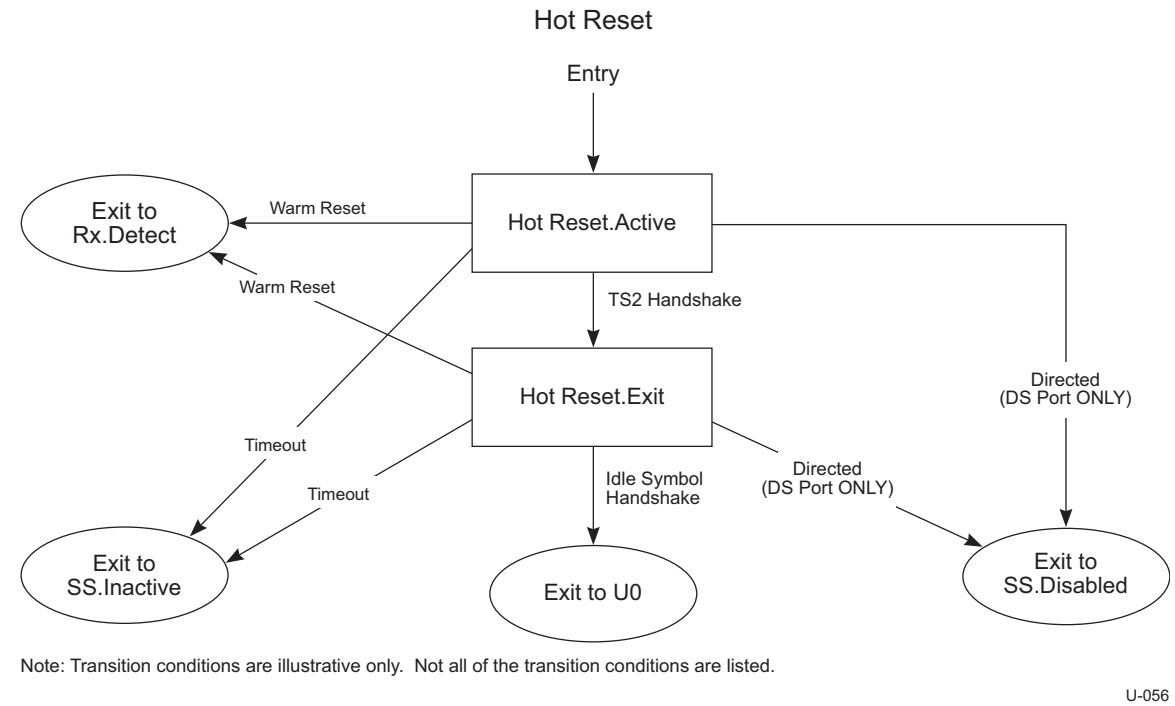


Figure 7-23. Hot Reset Substate Machine

8 Protocol Layer

The protocol layer manages the end to end flow of data between a device and its host. This layer is built on the assumption that the link layer guarantees delivery of certain types of packets and this layer adds on end to end reliability for the rest of the packets depending on the transfer type.

The chapter describes the following in detail:

- Types of packets
- Format of the packets
- Expected responses to packets sent by the host and a device
- The four SuperSpeed transaction types
- Support for Streams for the bulk transfer type
- Timing parameters for the various responses and packets the host or a device may receive or transmit

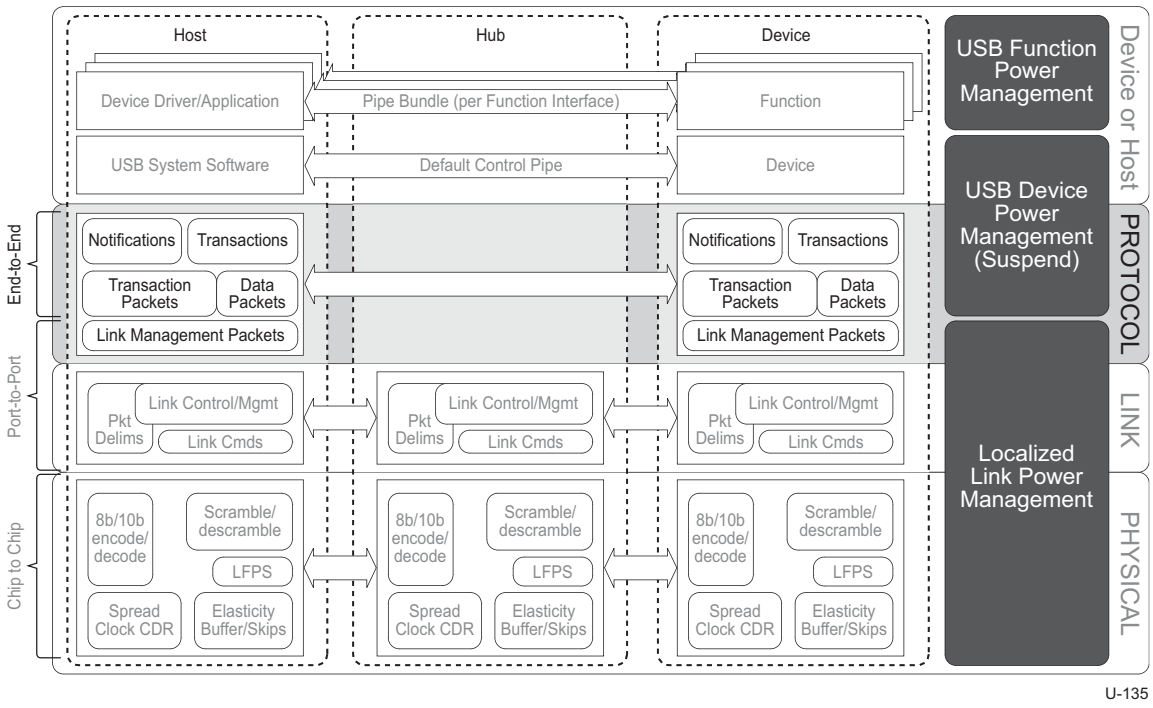


Figure 8-1. Protocol Layer Highlighted

8.1 SuperSpeed Transactions

SuperSpeed transactions are initiated by the host when it either requests or sends data to an endpoint on a device and are completed when the endpoint sends the data or acknowledges receipt of the data. A transfer on the SuperSpeed bus is a request for data made by a device application to the host which then breaks it up into one or more burst transactions. A SuperSpeed host may initiate one or more OUT bus transactions to one or more endpoints while it waits for the completion of the current bus transaction. However a SuperSpeed host shall not initiate another IN bus transaction to any endpoint until the host either:

- Receives all DPs or an NRDY or a STALL TP or the transaction times out for the current ACK TP sent to a non-isochronous endpoint or
- Receives all the DPs that were requested or it receives a short packet or it receives a DP with last packet flag field set or the transaction times out for the current ACK TP sent to an isochronous endpoint.

For non-isochronous transfers, an endpoint may respond to valid transactions by:

- Returning an NRDY Transaction Packet
- Accepting it by returning an ACK Transaction Packet in the case of an OUT transaction
- Returning one or more data packets in the case of an IN transaction
- Returning a STALL Transaction Packet if there is an internal endpoint error

An NRDY Transaction Packet (TP) response indicates that an endpoint is not ready to sink or source data. This allows the links between the device and the host to be placed in a reduced power state until an endpoint is ready to receive or send data. However, as mentioned in Section 8.10.1, the host may continue to perform transactions with the endpoint on the device even before the endpoint notifies the host that it is ready. This allows the links between the device and the host to be placed in a reduced power state until an endpoint is ready to receive or send data. When ready, the endpoint asynchronously sends a notification (ERDY TP) to the host to tell it that it is now ready to move data and the host responds by rescheduling the request. Note that isochronous transfers do not use ERDY or NRDY TPs as they are serviced by the host at periodic intervals. Additionally, data packets sent to or received from an isochronous endpoint are not acknowledged, i.e., no ACK TPs are sent to acknowledge the receipt of data packets.

Endpoints only respond to requests made by the host. The host is responsible for scheduling transactions on the bus and maintaining the priority and fairness of the data movement on the bus; it does this by the timing and ordering of IN and OUT requests. Transactions are not broadcast; packets traverse a direct path between the host and device. Any unused links may be placed into reduced power states making the bus amenable to aggressive power management.

8.2 Packet Types

SuperSpeed USB uses four basic packet types each with one or more subtypes. The four packet types are:

- **Link Management Packets** (LMP) only travel between a pair of links (e.g., a pair of directly connected ports) and is primarily used to manage that link.
- **Transaction Packets** (TP) traverse all the links directly connecting the host to a device. They are used to control the flow of data packets, configure devices, and hubs, etc. Transaction Packets have no data payload.
- **Data Packets** (DP) traverse all the links directly connecting the host to a device. Data Packets have two parts: a Data Packet Header (DPH) and a Data Packet Payload (DPP).
- **Isochronous Timestamp Packets** (ITP) are multicast on all the active links from the host to one or more devices.

All packets consist of a 14-byte header, followed by a 2-byte Link Control Word at the end of the packet (16 bytes total). All headers have a Type field that is used by the receiving entity (e.g., host, hub, or device) to determine how to process the packet. All headers include a 2-byte CRC (CRC-16). Packet headers have an uncorrectable or undetectable error rate less than one error in 10^{20} bits.

All devices (including hubs) and the host consume the LMPs they receive. Hubs have the additional responsibility to forward DPs, ITPs, and TPs to the downstream port nearer a device or to the upstream port nearer the host. Note that ITPs are only sent by the host and received by devices. All packets except LMPs are forwarded by hubs unless the packet is routed to the hub itself. Additional rules for forwarding ITPs are described in Section 8.7. Note that the Link Control Word in a TP, ITP, or DPH may be changed by a hub before it is forwarded. The fields in the Link Control Word are described in Section 8.3.1.2.

If the value of the **Type** field is *Transaction Packet* or *Data Packet Header*, the **Route String** and **Device Address** fields follow the **Type** field. The **Route String** field is used by hubs to route packets which appear on their upstream port to the appropriate downstream port. Packets flowing from a device to the host are always routed from a downstream port on a hub to its upstream port. The **Device Address** field is provided to the host so that it can identify the source of a packet. All other fields are discussed further in this chapter.

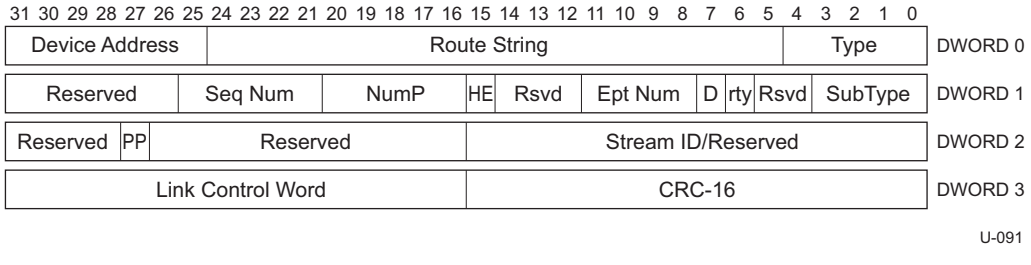


Figure 8-2. Example Transaction Packet

Data Packets include additional information in the header that describes the data block. The Data Block is always followed by a 4-byte CRC-32 used to determine the correctness of the data. The Data Block and the CRC-32 together are referred to as the Data Packet Payload.

8.3 Packet Formats

This section defines the SuperSpeed packets. It defines the fields that make up the various packet type and subtypes.

Packet byte and bit definitions in this section are described in an un-encoded data format. The effects of symbols added to the serial stream (i.e., to frame packets or control or modify the link), bit encoding, bit scrambling, and link level framing have been removed for the sake of clarity. Refer to Chapters 6 and 7 for detailed information. In cases where bus performance, efficiency, or timing are discussed, the effects of these lower level operations will be discussed to provide additional context.

8.3.1 Fields Common to all Headers

All SuperSpeed headers start with the **Type** field that is used to determine how to interpret the packet. At a high level this tells the recipient of the packet what to do with it: either to use it to manage the link or to move and control the flow of data between a device and the host.

8.3.1.1 Reserved Values and Reserved Field Handling

Reserved fields and Reserved values shall not be used in a vendor-specific manner.

A transmitter shall set all **Reserved** fields to zero and a receiver shall ignore any **Reserved** field.

A transmitter shall not set a defined field to a reserved value and a receiver shall ignore any packet that has any of its defined fields set to a reserved value. Note that the receiver shall acknowledge the packet and return credit for the same as per the requirement specified in Section 7.2.4.1.

8.3.1.2 Type Field

The **Type** field is a 5-bit field that identifies the format of the packet. The type is used to determine how the packet is to be used or forwarded by intervening links.

Table 8-1. Type Field Description

Width (bits)	Offset (DW:bit)	Description
5	0:0	Type. These 5 bits identify the packet's Type. **Value** — **Description** 00000b — Link Management Packet. 00100b — Transaction Packet 01000b — Data Packet Header 01100b — Isochronous Timestamp Packet All other values are Reserved.

8.3.1.3 CRC-16

All header packets have a 16-bit CRC field. This field is the CRC calculated over the preceding 12 bytes in the header packet. Refer to Section 7.2.1.1.2 for the polynomial used to calculate this value.

8.3.1.4 Link Control Word

The usage of the Link Control Word is defined in Section 7.2.1.1.3.

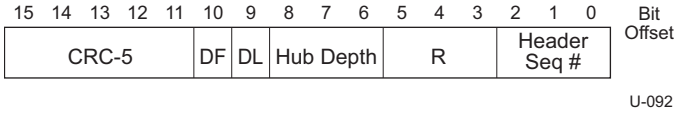


Figure 8-3. Link Control Word Detail

Table 8-2. Link Control Word Format

Width (bits)	Offset (DW:bit)	Description
3	3:16	**Header Sequence Number**. The valid values in this field are 0 through 7.
3	3:19	**Reserved (R)**.
3	3:22	**Hub Depth**. This field is only valid when the **Deferred** bit is set and identifies to the host the hierarchical on the USB that the hub is located at in the deferred TP or DPH returned to the host. This informs the host that the port on which the packet was supposed to be forwarded on is currently in a low power state (either U1 or U2). The only valid values in this field are 0 through 4.
1	3:25	**Delayed (DL)**. This bit shall be set if a Header Packet is resent or the transmission of a Header Packet is delayed. Chapter 7 and Chapter 10 provide more details on when this bit shall be set. This bit shall not be reset by any subsequent hub that this packet traverses.
1	3:26	**Deferred (DF)**. This bit may only be set by a hub. This bit shall be set when the downstream port on which the packet needs to be sent is in a power managed state. This bit shall not be reset by any subsequent hub that this packet traverses.
5	3:27	**CRC-5**. This field is the CRC used to verify the correctness of the preceding 11 bits in this word. Refer to Section 7.2.1.1.3 for the polynomial used to calculate this value.

8.4 Link Management Packet (LMP)

Packets that have the **Type** field set to *Link Management Packet* are referred to as LMPs. These packets are used to manage a single link. They carry no addressing information and as such are not routable. They may be generated as the result of hub port commands. For example, a hub port command is used to set the U2 inactivity timeout. In addition, they are used to exchange port capability information and testing purposes.

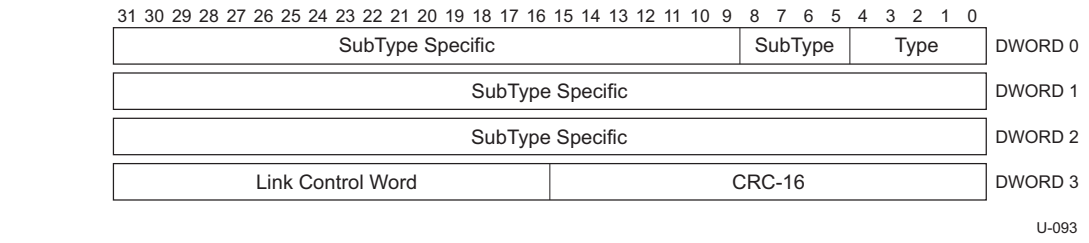


Figure 8-4. Link Management Packet Structure

8.4.1 Subtype Field

The value in the LMP **Subtype** field further identifies the content of the LMP.

Table 8-3. Link Management Packet Subtype Field

Width (bits)	Offset (DW:bit)	Description
4	0:5	**Subtype**. These 4 bits identify the Link Packet Subtype. **Value** — **Type of LMP** 0000b — Reserved 0001b — Set Link Function 0010b — U2 Inactivity Timeout 0011b — Vendor Device Test 0100b — Port Capability 0101b — Port Configuration 0110b — Port Configuration Response 0111b-1111b — Reserved
16	3:0	**CRC-16**. This field is the CRC calculated over the preceding 12 bytes. Refer to Section 7.2.1.1.2 for the polynomial used to calculate this value.

8.4.2 Set Link Function

The **Set Link Function** LMP shall be used to configure functionality that can be changed without leaving the active (U0) state.

Upon receipt of an LMP with the Force_LinkPM_Accept bit asserted, the port shall accept all LGO_U1 and LGO_U2 Link Commands until the port receives an LMP with the Force_LinkPM_Accept bit de-asserted. After port receives an LMP with the Force_LinkPM_Accept bit de-asserted, port will function in normal mode doing power management based on packet pending state of device's endpoints.

The device must stay in U1 or U2 until the downstream port initiates exit to U0. Software must ensure that there are no pending packets at the link level before issuing a SetPortFeature command that generates an LGO_U1 or LGO_U2 link command.

During normal operation, this feature shall only be used if all other means of lowering the link state from U0 to U1 or U2 fail.

This LMP is sent by a hub to a device connected on a specific port when it receives a SetPortFeature (FORCE_LINKPM_ACCEPT) command. Refer to Section 10.14.2.2 and Section 10.14.2.10 for more details.

Note: Improper use of the Force_LinkPM_Accept functionality can impact the performance of the link significantly and in some cases (when used during normal operation only) may lead to the device being unable to return to proper operation.

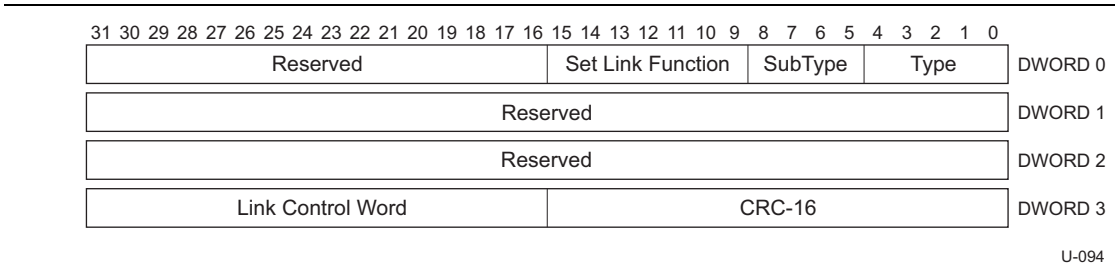


Figure 8-5. Set Link Function LMP

Table 8-4. Set Link Function

Width (bits)	Offset (DW:bit)	Description
4	0:5	**Subtype**. This field shall be set to *Set Link Function* for a Set Link Function LMP.
7	0:9	**Set Link Function**. These 7 bits identify the Set Link Function. **Bits** — **Description** 0 — Reserved. 1 — Force_LinkPM_Accept **Value** — **Meaning** 0 — De-assert 1 — Assert 6:2 — Reserved.

8.4.3 U2 Inactivity Timeout

The U2 Inactivity Timeout LMP shall be used to define the timeout from U1 to U2. Refer to Section 10.4.2.1 for details on this LMP.

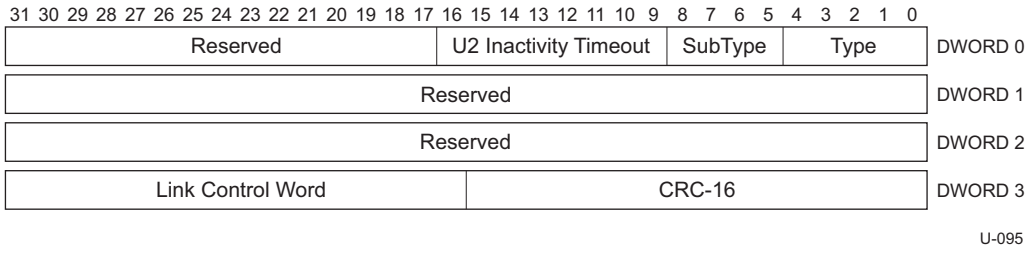


Figure 8-6. U2 Inactivity Timeout LMP

Table 8-5. U2 Inactivity Timer Functionality

Width (bits)	Offset (DW:bit)	Description
4	0:5	**Subtype**. This field shall be set to *U2 Inactivity Timeout* for a U2 Inactivity Timeout LMP.
8	0:9	**U2 Inactivity Timeout**. These 8 bits represent the U2 Inactivity Timeout value. The value placed in this field is the same value that is sent to the hub in a Set Port Feature (PORT_U2_TIMEOUT) command. Refer to Section 10.14.2.9 for details on the encoding of this field.

8.4.4 Vendor Device Test

Use of this LMP is intended for vendor-specific device testing and shall not be used during normal operation of the link.

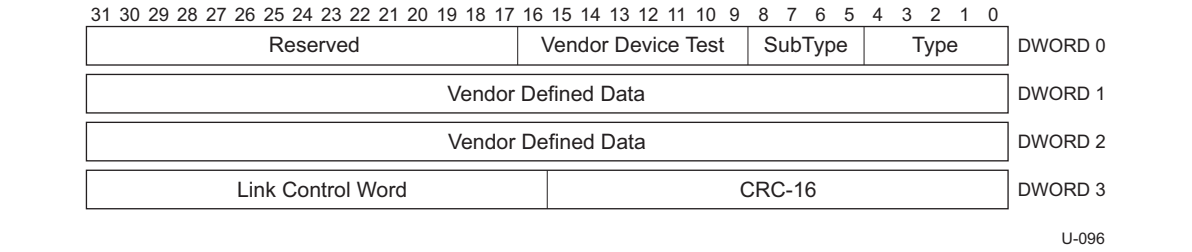


Figure 8-7. Vendor Device Test LMP

Table 8-6. Vendor-specific Device Test Function

Width (bits)	Offset (DW:bit)	Description
4	0:5	**Subtype**. This field shall be set to *Vendor Device Test*.
8	0:9	**Vendor-specific device test**. The function of these 8 bits is vendor specific.
64	1:0	**Vendor-defined data**. This value is vendor-defined.

8.4.5 Port Capabilities

The Port Capability LMP describes each port's link capabilities and is sent by both link partners after the successful completion of training and link initialization. After the port enters U0 from Polling, the port shall send Port Capability LMP within tPortConfiguration time once link initialization (refer to Section 7.2.4.1.1) is completed. Note the port may not always transition directly from Polling to U0, but may transition through other intermediate states (e.g., Recovery or Hot Reset) before entering U0. Regardless of states passed through between Polling and entry into U0, the device shall send a Port Capability LMP immediately upon entering U0.

If a link partner does not receive this LMP within tPortConfiguration time then:

- If the link partner has downstream capability, it shall signal an error as described in Section 10.14.2.6.
- If the link partner only supports upstream capability, then the upstream port shall transition to SS.Disabled and it shall try to connect at the other speeds this device supports.

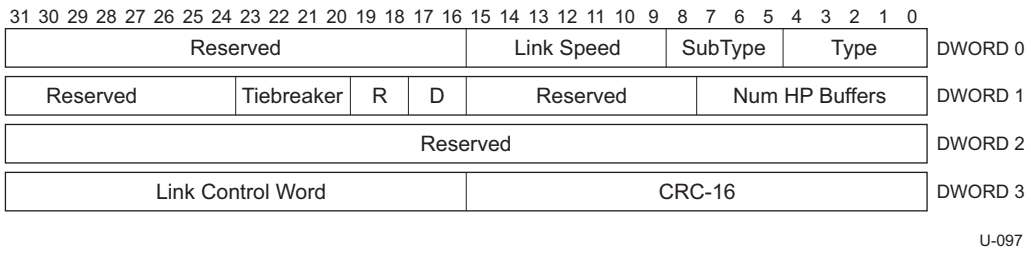


Figure 8-8. Port Capability LMP

Table 8-7. Port Capability LMP Format

Width (bits)	Offset (DW:bit)	Description
4	0:5	**SubType.** This field shall be set to *Port Capability*.
7	0:9	**Link speed**. This field is a bitmask that describes the link speeds supported by this device. **Bits** — **Description** 0 — This bit shall be set to 1 to indicate this device supports signaling at 5 Gbps 6:1 — **Reserved.**
16	0:16	**Reserved (R).**
8	1:0	**Num HP Buffers**. This field specifies the number of header packet buffers (in each direction Transmit or Receive) this device supports. All devices that are compliant to this revision of the specification shall return a value of 4 in this field. All other values are reserved.
8	1:8	**Reserved (R)**.
2	1:16	**Direction (D)**. This field is used to identify the upstream or downstream capabilities of the port. All ports shall have at least one of these bits set. **Bits** — **Description** 0 — If this bit is set to 1, then this port can be configured to be a downstream port. 1 — If this bit is set to 1, then this port can be configured to be an upstream port.
1	1:18	**USB 3.0 OTG Capable (OTG).** This field shall be set to 1 if the port supports the OTG Capability. Refer to Section 6.4 of the *On-The-Go and Embedded Host Supplement to the USB 3.0 Specification (Revision 3.0)* for more information.
1	1:19	**Reserved (R)**.
4	1:20	**Tiebreaker.** This field is only valid when both bits 0 and 1 of the **Direction** field are set. This field is used to determine the port type when two devices with both upstream and downstream capability are connected to each other. See Table 8-8 for details. This field shall be set to zero in all other cases.
40	1:24	**Reserved**.

After exchanging Port Capability LMPs, the link partners shall determine which of the link partners shall be configured as the downstream facing port as specified in Table 8-8.

Table 8-8. Port Type Selection Matrix

		Port 2		
		Upstream Only	**Downstream Only**	**Both**
Port 1	**Upstream Only**	Not Defined	Port 2 is the downstream port.	Port 2 is the downstream port.
	Downstream Only	Port 1 is the downstream port.	Not Defined	Port 1 is the downstream port.
	Both	Port 1 is the downstream port.	Port 2 is the downstream port.	The port with the higher value in the **Tiebreaker** field shall become the downstream port[1].

Note: [1]If the **TieBreaker** field contents are equal, then the two link partners shall exchange Port Capability LMPs again with new and different value in the **TieBreaker** field. The sequence of TieBreaker field values generated by a port shall be sufficiently random.

8.4.6 Port Configuration

Only the fields that are different from the Port Capability LMP are described in this section.

All SuperSpeed ports that support downstream port capability shall be capable of sending this LMP.

If the port that was to be configured in the upstream facing mode does not receive this LMP within tPortConfiguration time after link initialization, then the upstream port shall transition to SS.Disabled and a peripheral device shall try and connect at the other speeds this device supports.

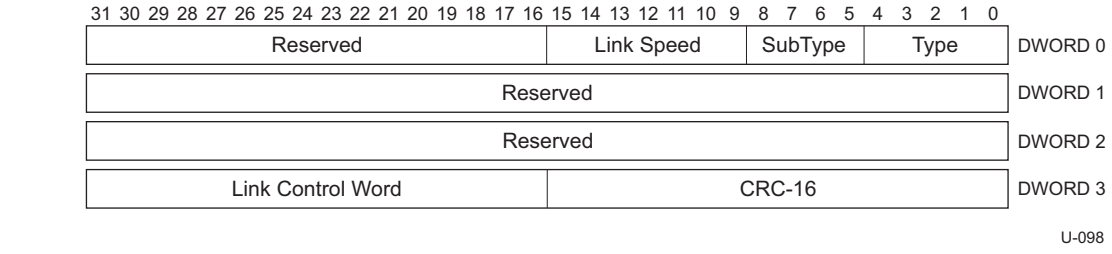


Figure 8-9. Port Configuration LMP

Table 8-9. Port Configuration LMP Format (Differences with Port Capability LMP)

Width (bits)	Offset (DW:bit)	Description
4	0:5	**SubType.** This field shall be set to *Port Configuration.*
7	0:9	**Link speed**. This field describes the link speed at which the upstream port shall operate. Only one of the bits in this field shall be set in the Port Configuration LMP sent by the link partner configured in the downstream mode. **Bits** — **Description** 0 — If this bit is set to 1, then this device shall operate at 5 Gbps. 6:1 — Reserved.
80	0:16b	**Reserved**.

A port configured in the downstream mode shall send the Port Configuration LMP to the upstream port. The port sending this LMP shall select only one bit for the **Link Speed** field.

If a downstream capable port cannot work with its link partner, then the downstream capable port shall signal an error as described in Section 10.14.2.6.

8.4.7 Port Configuration Response

This LMP is sent by the upstream port in response to a Port Configuration. It is used to indicate acceptance or rejection of the Port Configuration LMP. Only the fields that are different from the Port Capability LMP are described in this section.

All SuperSpeed ports that support upstream port capability shall be capable of sending this LMP.

If the downstream port does not receive this LMP within tPortConfiguration time, it shall signal an error as described in Section 10.14.2.6.

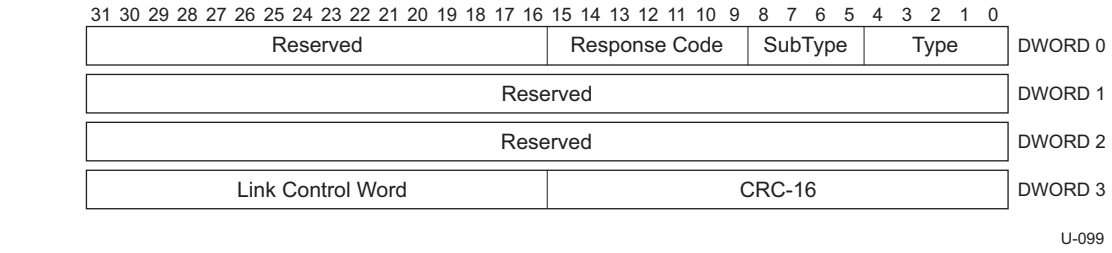


Figure 8-10. Port Configuration Response LMP

Table 8-10. Port Configuration Response LMP Format (Differences with Port Capability LMP)

Width (bits)	Offset (DW:bit)	Description
4	0:5	**SubType.** This field shall be set to *Port Configuration Response.*
7	0:9	**Response Code**. This field indicates the settings that were accepted in the Port Configuration LMP that was sent to a device. **Bits** — **Description** 0 — If this bit is set to 1, then this device accepted the Link Speed setting. 6:1 — Reserved.
80	0:16	**Reserved**.

If the Response Code indicates that the Link Speed was rejected by the upstream port, the downstream port shall signal an error as described in Section 10.14.2.6.

8.5 Transaction Packet (TP)

Transaction Packets (TPs) traverse the direct path between the host and a device. TPs are used to control data flow and manage the end-to-end connection. The value in the **Type** field shall be set to *Transaction Packet*. The **Route String** field is used by hubs to route a packet that appears on its upstream port to the correct downstream port. The route string is set to zero for a TP sent by a device. When the host sends a TP, the **Device Address** field contains the address of the intended recipient. When a device sends a TP to the host then it sets the **Device Address** field to its own address. This field is used by the host to identify the source of the TP. The **SubType** field in a TP is used by the recipient to determine the format and usage of the TP.

Table 8-11. Transaction Packet Subtype Field

Width (bits)	Offset (DW:bit)	Description
4	1:0	**Subtype**. The subtype field is used to identify a specific type of TP. **Value** — **Type of TP** 0000b — Reserved 0001b — ACK 0010b — NRDY 0011b — ERDY 0100b — STATUS 0101b — STALL 0110b — DEV_NOTIFICATION 0111b — PING 1000b — PING_RESPONSE 1001b – 1111b — Reserved

8.5.1 Acknowledgement (ACK) Transaction Packet

This TP is used for two purposes:

- For IN endpoints, this TP is sent by the host to request data from a device as well as to acknowledge the previously received data packet.
- For OUT endpoints, this TP is sent by a device to acknowledge receipt of the previous data packet sent by the host, as well as to inform the host of the number of data packet buffers it has available after receipt of this packet.

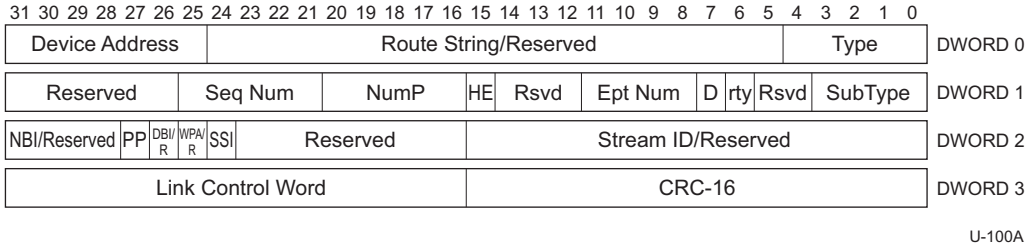


Figure 8-11. ACK Transaction Packet

Table 8-12. ACK TP Format

Width (bits)	Offset (DW:bit)	Description
20	0:5	**Route String/Reserved**. This field is only used by hubs. In conjunction with the hub depth, it is used to route a packet to the correct downstream port. Refer to Section 8.9 for details. When sent by a device, this field is Reserved.
7	0:25	**Device Address**. This field specifies the device, via its address, that is the recipient or the source of the TP. Refer to Section 8.8.
4	1:0	**SubType**. This field shall be set to *ACK* for an ACK TP.
2	1:4	**Reserved (Rsvd)**.
1	1:6	**Retry Data Packet (rty)**. This field is used to signal that the host or a device did not receive a data packet or received a corrupted data packet and requests the transmitter to resend one or more data packets starting at the specified sequence number.
1	1:7	**Direction (D)**. This field defines the direction of an endpoint within the device that is the source or recipient of this TP. Refer to Section 8.8. **Value** — **Direction of Data Flow** 0b — Host to Device 1b — Device to Host
4	1:8	**Endpoint Number (Ept Num)**. This field determines an endpoint within the device that is the source or recipient of this TP. Refer to Section 8.8.
3	1:12	**Reserved (Rsvd)**.
1	1:15	**Host Error (HE)**. This field is only valid when the ACK TP is sent from the host to a device. This bit shall be set if the host was unable to accept a valid data packet due to internal host issues. When the host sets this field, it must also set the **Retry Data Packet** field for a non-isochronous transfer.
5	1:16	**Number of Packets (NumP)**. This field is used to indicate the number of Data Packet buffers that the receiver can accept. The value in this field shall be less than or equal to the maximum burst size supported by the endpoint as determined by the value in the **Burst Size** field in the Endpoint Companion Descriptor (refer to Section 9.6.7).
5	1:21	**Sequence Number (Seq Num)**. This field is used to identify the sequence number of the next expected data packet.
6	1:26	**Reserved**.
16	2:0	**Stream ID/Reserved**. If this ACK TP is targeted at a Bulk endpoint that supports Streams (i.e., a Stream pipe), this field contains a **Stream ID** value between 1 and 65535. The **Stream ID** value of 0 is reserved for Stream pipes and the TP shall be considered invalid if a 0 value is received. All other pipe types shall treat this field as reserved. The usage of this field is class dependent. This field shall be set to zero if the Bulk endpoint does not support Streams. Refer to Section 8.12.1.4 for more information on Stream IDs.
8	2:16	**Reserved**.

Width (bits)	Offset (DW:bit)	Description
1	2:24	**Support Smart Isochronous (SSI).** This field is only valid for ISO endpoints. For OUT endpoints, the **DBI**, **WPA** and **NBI** fields are valid only if this field is set to one and the **Ipf** field is set to zero. In the case of IN endpoints, the host does not set the **Ipf** field and hence it just has to set the SSI field if the host supports smart isochronous scheduling. It informs the device that this host controller supports advanced isochronous scheduling functionality that can be used by the device to drive its link to lower power states in between the times that the host is polling the ISO endpoint in its service interval. In the case the host is transferring data to an OUT endpoint, it is the responsibility of the host controller that it only sets this field to one when the **Ipf** field is set to zero. Device response when both these fields are set to one is undefined. In the case the host is transferring data from an IN endpoint, then if the device responds with a DP that has the **Ipf** field set to one, then it can ignore the value in this field (and other related fields) and simply wait for the host to PING the device again before the endpoint is serviced
1	2:25	**Will Ping Again (WPA/Reserved).** This field is only valid for ISO endpoints and is only valid when the **SSI** field is set to one. If this field is set to one, then the host controller will send a PING TP before it services the endpoint again.
1	2:26	**Data in this Bus Interval is done (DBI/Reserved).** This field is only valid for ISO endpoints and is only valid when the **SSI** field is set to one. If this field is set to one, then the host controller is done performing transactions to this endpoint in the current bus interval. Note: WPA has a higher priority than this field. When a host sets the WPA field, the device can safely ignore the value in this field as the host will PING the device before resuming transactions to this endpoint.
1	2:27	**Packets Pending (PP)**. This field can only be set by the Host. If the field is set, then the host is ready to receive another DP from this endpoint/Stream. Where the endpoint is identified by the **Endpoint Number** and **Direction** fields, and if this is a Stream endpoint, then the Stream is identified by the **Stream ID** field. If this field is cleared, then the host is not ready to receive any more DPs for this Endpoint/Stream. If no endpoints on this device have packets pending, then the device can use this information to aggressively power manage its upstream link, e.g., set the link to a lower power U1 or U2 state.
4	2:28	**Number of Bus Intervals (NBI/Reserved).** This field is only valid for ISO endpoints and is only valid when the **SSI** field is set to one, the **WPA** field is set to zero and the **DBI** field is set to one. The value in this field informs the device the number of bus intervals after which the host controller will perform transactions to the endpoint again. The value in this field indicates to the endpoint that the host controller will service the endpoint in the bus interval with a value equal to (current bus interval + value in **NBI** field + 1).

8.5.2 Not Ready (NRDY) Transaction Packet

This TP can only be sent by a device for a non-isochronous endpoint. An OUT endpoint sends this TP to the host if it has no packet buffer space available to accept the DP sent by the host. An IN endpoint sends this TP to the host if it cannot return a DP in response to an ACK TP sent by the host.

Only the fields that are different from an ACK TP are described in this section.

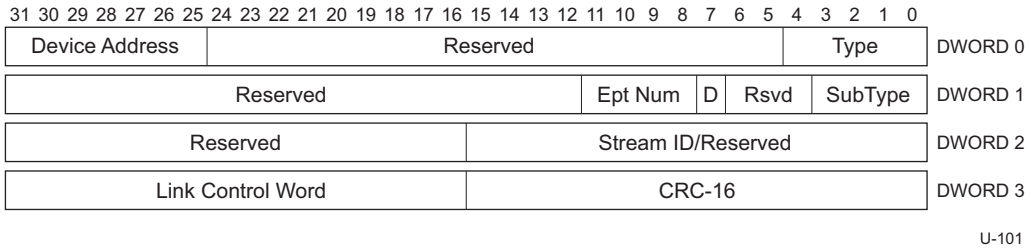


Figure 8-12. NRDY Transaction Packet

Table 8-13. NRDY TP Format (Differences with ACK TP)

Width (bits)	Offset (DW:bit)	Description
4	1:0	**SubType**. This field shall be set to *NRDY*.
3	1:4	**Reserved (Rsvd)**.
20	1:12	**Reserved**.
5	2:27	**Reserved**.

8.5.3 Endpoint Ready (ERDY) Transaction Packet

This TP can only be sent by a device for a non-isochronous endpoint. It is used to inform the host that an endpoint is ready to send or receive data packets. Only the fields that are different from an ACK TP are described in this section.

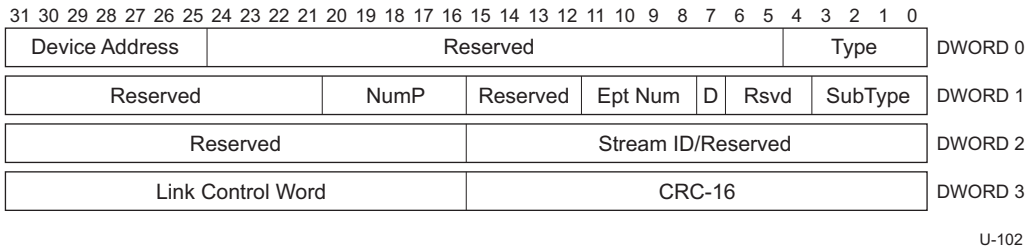


Figure 8-13. ERDY Transaction Packet

Table 8-14. ERDY TP Format (Differences with ACK TP)

Width (bits)	Offset (DW:bit)	Description
4	1:0	**SubType**. This field shall be set to *ERDY*.
3	1:4	**Reserved (Rsvd)**.
4	1:12	**Reserved**.
5	1:16	**Number of Packets (NumP)**. For an OUT endpoint, refer to Table 8-12 for the description of this field. For an IN endpoint this field is set by the endpoint to the number of packets it can transmit when the host resumes transactions to it. This field shall not have a value greater than the maximum burst size supported by the endpoint as indicated by the value in the **Burst Size** field in the Endpoint Companion Descriptor. Note that the value reported in this field may be treated by the host as informative only.
11	1:21	**Reserved**.
5	2:27	**Reserved**.

8.5.4 STATUS Transaction Packet

This TP can only be sent by the host. It is used to inform a control endpoint that the host has initiated the Status stage of a control transfer. This TP shall only be sent to a control endpoint. Only the fields that are different from an ACK TP are described in this section.

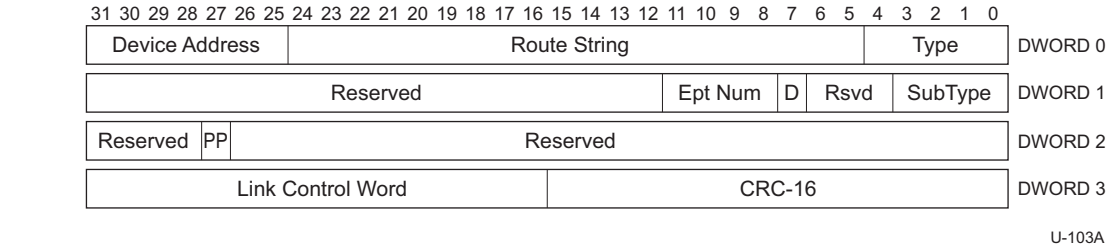


Figure 8-14. STATUS Transaction Packet

Table 8-15. STATUS TP Format (Differences with ACK TP)

Width (bits)	Offset (DW:bit)	Description
4	1:0	**SubType**. This field shall be set to *STATUS*.
3	0:4	**Reserved (Rsvd)**.
52	1:12	**Reserved**.

8.5.5 STALL Transaction Packet

This TP can only be sent by an endpoint on the device. It is used to inform the host that the endpoint is halted or that a control transfer is invalid. Only the fields that are different from an ACK TP are described in this section.

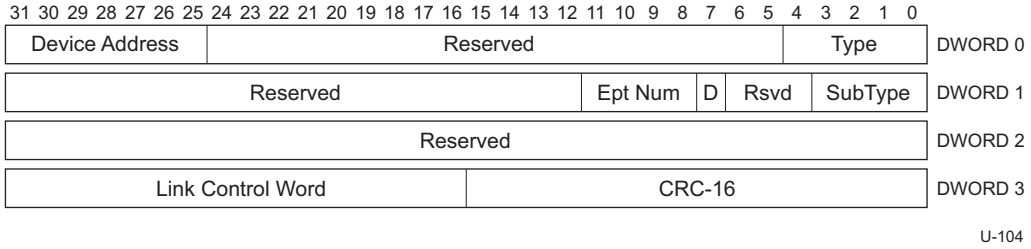


Figure 8-15. STALL Transaction Packet

Table 8-16. STALL TP Format (Differences with ACK TP)

Width (bits)	Offset (DW:bit)	Description
4	1:0	**SubType**. This field shall be set to *STALL*.
3	1:4	**Reserved (Rsvd)**.
52	1:12	**Reserved**.

8.5.6 Device Notification (DEV_NOTIFICATION) Transaction Packet

This TP can only be sent by a device. It is used by devices to inform the host of an asynchronous change in a device or interface state, e.g., to identify the function within a device that caused the device to perform a remote wake operation. This TP is not sent from a particular endpoint but from the device in general. Only the fields that are different from an ACK TP are described in this section.

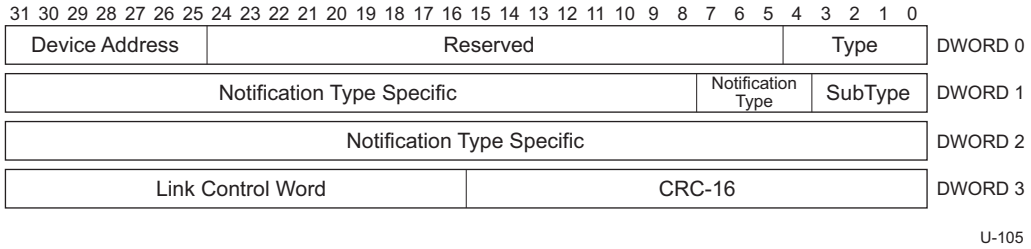


Figure 8-16. Device Notification Transaction Packet

Table 8-17. Device Notification TP Format (Differences with ACK TP)

Width (bits)	Offset (DW:bit)	Description
4	1:0	**SubType**. This field shall be set to *DEV_NOTIFICATION*.
4	1:4	**Notification Type**. The field identifies the type of the device notification. **Value** — **Type of Notification Packet** 0000b — Reserved 0001b — FUNCTION_WAKE 0010b — LATENCY_TOLERANCE_MESSAGE 0011b — BUS_INTERVAL_ADJUSTMENT_MESSAGE 0100b — HOST_ROLE_REQUEST[1] 0101b – 1111b — Reserved

8.5.6.1 Function Wake Device Notification

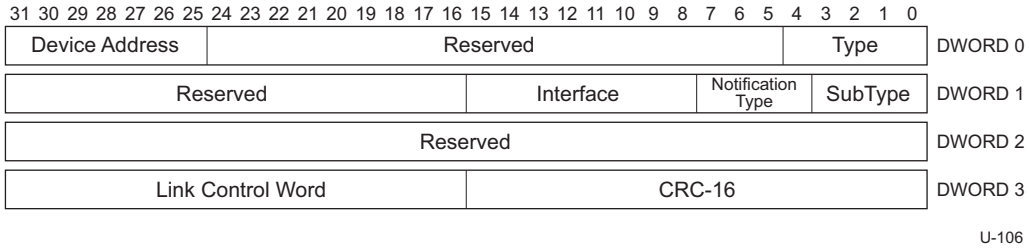


Figure 8-17. Function Wake Device Notification

Table 8-18. Function Wake Device Notification

Width (bits)	Offset (DW:bit)	Description
4	1:0	**SubType**. This field shall be set to *DEV_NOTIFICATION*.
4	1:4	**Notification Type**. FUNCTION_WAKE
8	1:8	**Interface**. This field identifies the first interface in the function that caused the device to perform a remote wake operation.
48	1:16	**Reserved**.

[1] This Notification Type value shall be reserved for OTG use. Refer to Section 6.4 of the USB 3.0 OTG and EH Supplement for the definition of the respective Link Management Packet (LMP).

8.5.6.2 Latency Tolerance Message (LTM) Device Notification

Latency Tolerance Message Device Notification is an optional normative feature enabling more power efficient platform operation.

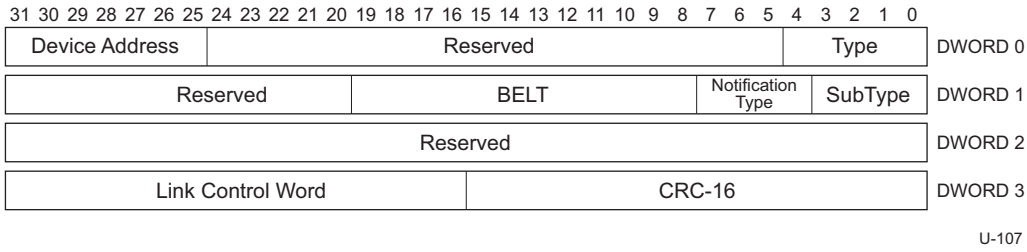


Figure 8-18. Latency Tolerance Message Device Notification

Table 8-19. Latency Tolerance Message Device Notification

Width (bits)	Offset (DW:bit)	Description
4	1:0	**SubType**. This field shall be set to *DEV_NOTIFICATION*.
4	1:4	**Notification Type**. LATENCY_TOLERANCE_MESSAGE.
12	1:8	**BELT**. This field describes the Best Effort Latency Tolerance value, representing the time in nanoseconds that a device can wait for service before experiencing unintended operational side effects. **Bits** **Description** 9:0 LatencyValue (ns) 11:10 LatencyScale **Value** / **Description** 00b – Reserved 01b – LatencyValue is to be multiplied by 1024 10b – LatencyValue is to be multiplied by 32,768 11b – LatencyValue is to be multiplied by 1,048,576
44	1:20	**Reserved**.

8.5.6.3 Bus Interval Adjustment Message Device Notification

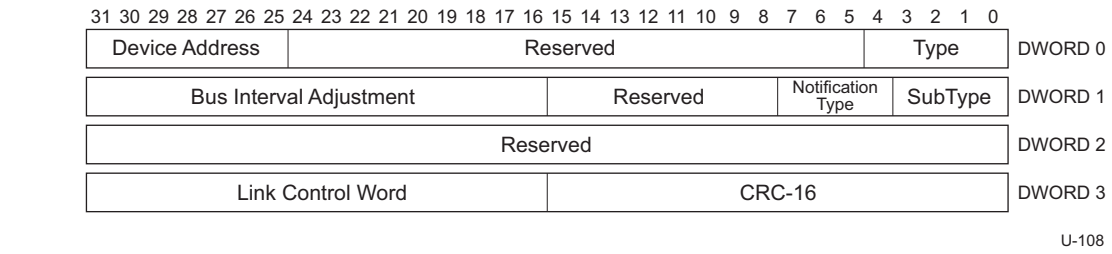


Figure 8-19. Bus Interval Adjustment Message Device Notification

Table 8-20. Bus Interval Adjustment Message Device Notification

Width (bits)	Offset (DW:bit)	Description
4	1:0	**SubType**. This field shall be set to *DEV_NOTIFICATION*.
4	1:4	**Notification Type**. BUS_INTERVAL_ADJUSTMENT_MESSAGE.
8	1:8	**Reserved**.
16	1:16	**Bus Interval Adjustment.** This field is a two's complement value ranging from -32768 to +32767 expressed in BusIntervalAdjustmentGranularity units.

8.5.6.4 Function Wake Notification

A function may signal that it wants to exit from device suspend (after transitioning the link to U0) or function suspend by sending a Function Wake Device Notification to the host if it is enabled for remote wakeup. Refer to Section 9.2.5 for more details.

8.5.6.5 Latency Tolerance Messaging

Latency Tolerance Messaging is an optional normative USB power management feature that utilizes reported BELT (Best Effort Latency Tolerance) values to enable more power efficient platform operation.

The BELT value is the maximum time (factoring in the service needs of all configured endpoints) for leaving a device without service from the host. Specifically, the BELT value is the time between the host's receipt of an ERDY from a device, and the host's transmission of the response to the ERDY.

Devices indicate whether they are capable of sending LTM TPs using the **LTM Capable** field in the SUPERSPEED_USB Device Capability descriptor in the BOS descriptor (refer to Section 9.6.2). The LTM Enable (refer to Section 9.4.10) feature selector enables (or disables) an LTM capable device to send LTM TPs.

8.5.6.5.1 Optional Normative LTM and BELT Requirements

General Device Requirements

- LTM TPs shall be originated only by peripheral devices.
- LTM TPs apply to all endpoint types except for isochronous endpoints. For interrupt endpoints the BELT value only applies while the endpoint is in a flow control condition.
- Once a BELT value has been sent to the host by a device, all configured endpoints for that device shall expect to be serviced within the specified BELT time.
- A device shall send an LTM TP with a value of tBELTdefault in the **BELT** field in response to any change in state of LTM_Enable within the timing specified by tMinLTMStateChange.
- A device shall ensure that its BELT value is determined frequently enough that it is able to provide reasonable estimate of the device's service latency tolerance prior to its need to change BELT value. In addition, the following conditions shall be met:
  - The maximum number of LTM TPs is bounded by tBeltRepeat.
  - Each LTM TP shall have a different BELT value.
- The system shall default to a BELT of 1 ms for all devices (refer to Table 8-33).
- The minimum value for a BELT is 125 µs (refer to Table 8-33).

Device Requirements Governing Establishment of BELT Value

- The LTM mechanism shall utilize U1SEL and U2SEL to provide devices with system latency information (see Section 9.4.12 – Set SEL). In this context, the system latency is the time between when a device transmits an ERDY and when it will receive a transaction packet (type is direction-specific) from the host when the deepest allowed link state is U1 or U2. These values are used by the device to properly adjust their BELT value, factoring in their location within the USB link topology.
  - Devices that allow their link to enter U1, but not U2, shall subtract the U1 System Exit Latency (U1SEL) from its total latency tolerance and send the resultant value as the **BELT** field value in an LTM TP.
  - Devices that allow their link to enter U1 and U2, shall subtract U2SEL from its total latency tolerance and send the resulting value as the **BELT** field value in an LTM TP.

8.5.6.6 Bus Interval Adjustment Message

This device notification may be sent by a device to request an increase or decrease in the length of the bus interval. This would typically be used by a device trying to synchronize the host's bus interval clock with an external clock. Bus interval adjustment requests are relative to the current bus interval. For example, if a device requests an increase of one BusIntervalAdjustmentGranularity unit and then later requests an increase of two BusIntervalAdjustmentGranularity units the overall increase by the host would be three BusIntervalAdjustmentGranularity units.

The host shall support adjustments through an absolute range of -37268 to 37267 BusIntervalAdjustmentGranularity units. A device shall not request adjustments more than once every eight bus intervals. A device shall not send another bus interval adjustment request until it has waited long enough to accurately observe the effect of the previous bus interval adjustment request on the timestamp value in subsequent ITPs. A device shall not make a single BusIntervalAdjustment request for more than ±4096 units. A device may make multiple

BusIntervalAdjustment requests over time for a combined total of more than 4096 units. A device shall not request a bus interval adjustment unless the device received an ITP within the past 125 μs, the ITP contained a **Bus Interval Adjustment Control** field with a value equal to zero or the device's address and the device is in the Address or Configured state.

Only one device can control the bus interval length at a time. The host controller implements a first come first serve policy for handling bus interval adjustment requests as described in this section. When the host controller begins operation it shall transmit ITPs with the **Bus Interval Adjustment Control** field set to zero. When the host controller first receives a bus interval adjustment control request, it shall set the **Bus Interval Adjustment Control** field in subsequent ITPs to the address of the device that sent the request. The host shall ignore bus interval adjustment requests from all other devices once the **Bus Interval Adjustment Control** field is set to a non-zero address. If the controlling device is disconnected, the host controller shall reset the **Bus Interval Adjustment Control** field to zero. The host controller may provide a way for software to override default bus interval adjustment control field behavior and select a controlling device. The host controller shall begin applying bus interval adjustments within two bus intervals from when the bus interval adjustment request is received.

The smallest bus interval adjustment (one BusIntervalAdjustmentGranularity) requires the host to make an average adjustment of eight high speed bit times every 4096 bus intervals. The host is allowed to make this adjustment in a single bus interval such that the clock used to generate ITP times and bus interval boundaries does not need a period smaller than eight high speed bit times. The host shall make bus interval adjustments at regular intervals. When the host is required to make an average of one or more eight high speed bit time adjustments every 4096 bus intervals the adjustments shall be evenly distributed as defined by the following constraints:

- Intervals that contain one more eight high speed bit time adjustment than other intervals are referred to as maximum adjustment bus intervals.
- The number of eight high speed bit time adjustments made in any bus interval shall not be more than one greater than the number of high speed bit time adjustments made in any other bus interval.
- The distance in bus intervals between consecutive maximum adjustment bus intervals shall not vary by more than one bus interval.

The even distribution and average adjustment requirements for bus interval adjustments shall apply from one bus interval after a bus interval adjustment request is received by the host until the bus interval where a subsequent valid bus interval adjustment request is received by the host.

The following is an example of valid host behavior for a specific bus interval adjustment request. After power on, the host receives a bus interval adjustment request for a bus interval decrease of 10 BusIntervalAdjustmentGranularity units in bus interval X-1. The host controller uses a clock with a period of eight high speed bit times to drive a counter that produces timestamps and bus interval boundaries. The host controller adds an extra eight high speed bit time clock tick to its counter in each of the following bus intervals: X+409, X+819, X+1228, X+1638, X+2048, X+2457, X+2867, X+3276, X+3686, X+4096, X+4505,….

8.5.7 PING Transaction Packet

This TP can only be sent by the host. It is used by the host to transition all links in the path to a device back to U0 prior to initiating an isochronous transfer. Refer to Appendix C for details on the usage of this TP. Only the fields that are different from an ACK TP are described in this section.

A device shall respond to the PING TP by sending a PING_RESPONSE TP (refer to Section 0) to the host within the tPingResponse time (refer to Table 8-33). Note that the device shall not validate the EP_NUM and Direction fields and simply copy them to the respective fields in the PING_RESPONSE TP.

A device shall keep its link in U0 until it receives a subsequent packet from the host, or until the tPingTimeout time (refer to Table 8-33) elapses.

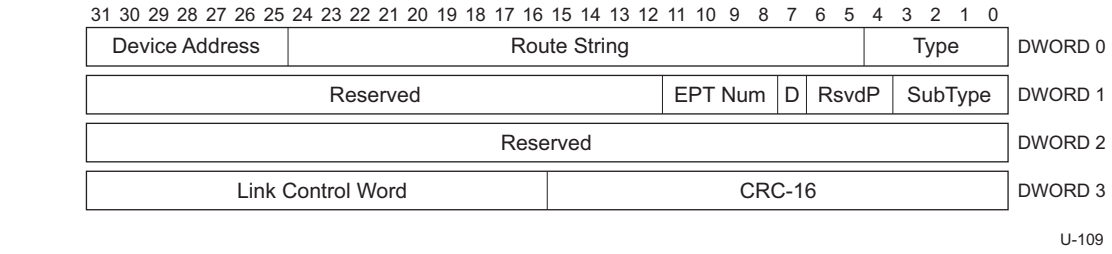


Figure 8-20. PING Transaction Packet

Table 8-21. PING TP Format (differences with ACK TP)

Width (bits)	Offset (DW:bit)	Description
4	1:0	**SubType**. This field shall be set to *PING*.
3	1:4	**Reserved**.
52	1:12	**Reserved**.

8.5.8 PING_RESPONSE Transaction Packet

This TP can only be sent by a device in response to a PING TP sent by the host. A PING_RESPONSE TP shall be sent for each PING TP received. Refer to Appendix C for details on the usage of this TP. Only the fields that are different from an ACK TP are described in this section.

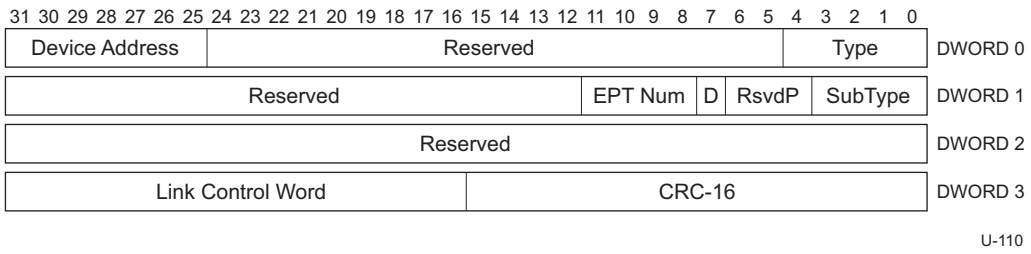


Figure 8-21. PING_RESPONSE Transaction Packet

Table 8-22. PING_RESPONSE TP Format (Differences with ACK TP)

Width (bits)	Offset (DW:bit)	Description
4	1:0	**SubType**. This field shall be set to *PING_RESPONSE*.
3	1:4	**Reserved**.
1	1:7	**Direction (D)**. This field shall be set to the value of the **Direction** field in the PING TP for which this PING_RESPONSE TP is being sent.
4	1:8	**Endpoint Number (Ept Num)**. This field shall be set to the value of the **Ept Num** field in the PING TP for which this PING_RESPONSE TP is being sent.
52	1:12	**Reserved**.

8.6 Data Packet (DP)

This packet can be sent by either the host or a device. The host uses this packet to send data to a device. Devices use this packet to return data to the host in response to an ACK TP. All data packets are comprised of a Data Packet Header and a Data Packet Payload. Only the fields that are different from an ACK TP are described in this section.

Data packets traverse the direct path between the host and a device. Note that it is permissible to send a data packet with a zero length data block; however, it shall have a CRC-32.

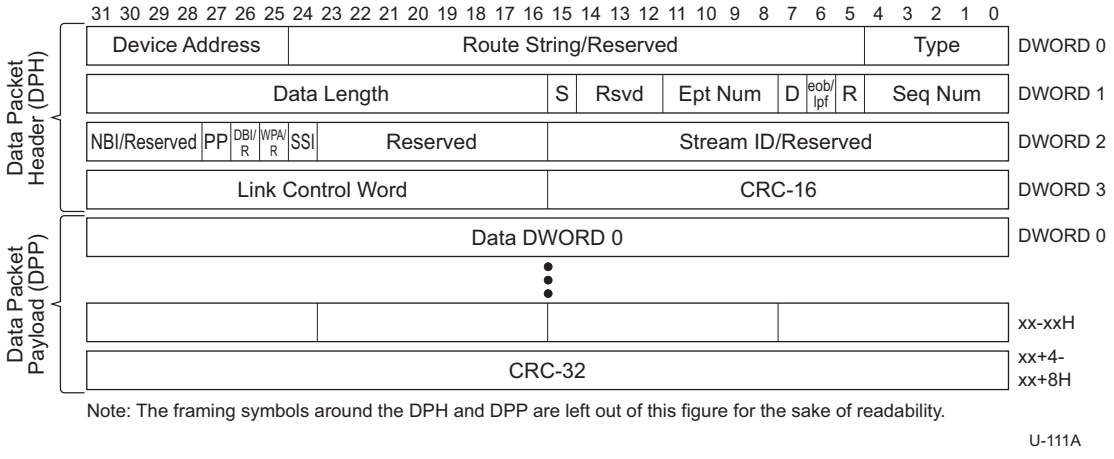


Figure 8-22. Example Data Packet

Table 8-23. Data Packet Format (Differences with ACK TP)

Width (bits)	Offset (DW:bit)	Description
5	1:0	**Sequence Number (Seq Num)**. This field is used to identify the sequence number of the DP. Note that the sequence number wraps around at 31.
1	1:5	**Reserved (R)**.
1	1:6	**End Of Burst (EOB)/Last Packet Flag (LPF)**. For non-isochronous endpoints, this field is referred to as EOB and for isochronous endpoints this field is referred to as LPF. For non-isochronous IN endpoints, this field is used to identify that this is the last packet of a burst. When a device is ready to continue the transfer, it shall send an ERDY TP to signal the host. Note that an endpoint shall re-evaluate the EOB value in a retried DP. The EOB field shall be set in the last packet of a burst if the device returns fewer than the number of packets requested in the NumP field of the last ACK TP it received and the last packet is not a short packet. Note that a device is not required to set this field to a 1b when it transmits a short packet even if it will be returning fewer than the number of packets requested in the NumP field of the last ACK TP it received. It is only required to set this field to a 1b if it wants to enter the flow control state after completing the current transfer with this short packet. For non-isochronous OUT and control endpoints, this field shall be set to zero. For isochronous endpoints this field is used to identify that this is the last packet of the last burst in the current service interval. LPF can be set by a device and the host. Please refer to Section 8.12.6 for the usage of this field when the target or source of this DP is an isochronous endpoint.
4	1:8	**Endpoint Number (Ept Num)**. This field determines an endpoint within the device that is the source or recipient of this DP.
3	1:12	**Reserved (R)**.
1	1:15	**Setup (S)**. This field is set by the host to indicate that this DP is a Setup data packet. This field can only be set by the host.
16b	1:16	**Data Length**. This field is used to indicate the number of bytes in the DPP excluding the data CRC-32.
1	2:27	**Packets Pending (PP).** This field may only be set by the Host. If this field is set, then the host has one or more DPs available for transmission to this endpoint/Stream. Where the endpoint is identified by the **Endpoint Number** and **Direction** fields, and if this is a Stream endpoint then the Stream is identified by the **Stream ID** field. If the field is cleared, then this is the last DP that the host has available for transmission to the target endpoint/Stream. If no endpoints on this device have packets pending, then the device can use this information to aggressively power manage its upstream link, e.g., set the link to a lower power U1 or U2 state.
xx	4:0	**Data Block**. This field contains the data in the DPP. The size of this field in bytes is indicated by the value in the **Data Length** field.
32	4:0 + xx	**CRC-32**. The data CRC is calculated over the data block of the DPP. Refer to Section 7.2.1.2.1 for the polynomial used to calculate this value. Note that this field is not necessarily aligned on a DWORD boundary as the data block length may not be a multiple of four.

8.7 Isochronous Timestamp Packet (ITP)

The value in the **Type** field is *Isochronous Timestamp Packet* for an ITP. ITPs are used to deliver timestamps from the host to all active devices. ITPs carry no addressing or routing information and are multicast by hubs to all of their downstream ports with links in the U0 state. A device shall not respond to an ITP. ITPs are used to provide host timing information to devices for synchronization. Note that any device or hub may receive an ITP. The host shall transmit an ITP on a root port link if and only if the link is already in U0. Only the host shall initiate an ITP transmission. The host shall not bring a root port link to U0 for the purpose of transmitting an ITP. The host shall transmit an ITP in every bus interval within tTimestampWindow from a bus interval boundary if the root port link is in U0. The host shall begin transmitting ITPs within tIsochronousTimestampStart from when the host root port's link enters U0 from the polling state. An ITP may be transmitted in between packets in a burst. If a device receives an ITP with the delayed flag (DL) set in the link control word, the timestamp value may be significantly inaccurate and may be ignored by the device.

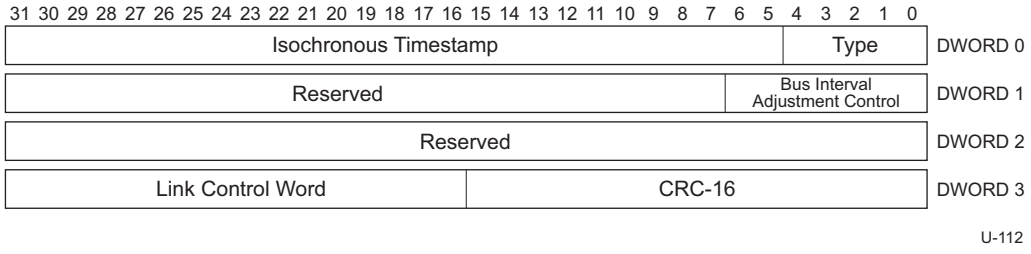


Figure 8-23. Isochronous Timestamp Packet

Table 8-24. Isochronous Timestamp Packet Format

Width (bits)	Offset (DW:bit)	Description
27	0:5	**Isochronous Timestamp (ITS)**. The isochronous timestamp field is used to identify the current time value from the perspective of the host transmitting the ITP. The timestamp field is split into two sub-fields: **Bits** — **Description** 13:0 — **Bus interval counter**. The current 1/8 of a millisecond counter. The count value rolls over to zero when the value reaches 0x3FFF and continues to increment. 26:14 — **Delta**. The time delta from the start of the current ITP packet to the previous bus interval boundary. This value is a number of tIsochTimestampGranularity units. The value used shall specify the delta that comes closest to the previous bus interval boundary without going before the boundary. Note: If a packet starts exactly on a bus interval boundary, the delta time is set to 0.
7	1:0	**Bus Interval Adjustment Control**. This field specifies the address of the device that controls the bus interval adjustment mechanism. Upon reset, power-up, or if the device is disconnected, the host shall set this field to zero.
57	1:7	**Reserved**.

The ITS value in the ITP shall have an accuracy of ±1 tIsochTimestampGranularity units of the value of the host clock (for ITP generation) measured when the first framing symbol of the ITP is transmitted by the host.

8.8 Addressing Triple

Data Packets and most Transaction Packets provide access to specific data flow using a composite of three fields. They are the **Device Address**, the **Endpoint Number**, and the **Direction** fields.

Upon reset and power-up, a device's address defaults to a value of zero and shall be programmed by the host during the enumeration process with a value in the range from 1 to 127. Device address zero is reserved as the default address and may not be assigned to any other use.

Devices may support up to a maximum of 15 IN and 15 OUT endpoints (as indicated by the **Direction** field) apart from the required default control endpoint that has an endpoint number set to zero.

8.9 Route String Field

The **Route String** is a 20-bit field in downstream directed packet that the hub uses to route the packet to the designated downstream port. It is composed of a concatenation of the downstream port numbers (4 bits per hub) for each hub traversed to reach a device. The hub uses a Hub Depth value multiplied by four as an offset into the Route String to locate the bits it uses to determine the downstream port number. The Hub Depth value is determined and assigned to every hub during the enumeration process.

Note that this field is only valid in packets sent by the host and when sent by a device, this field is Reserved.

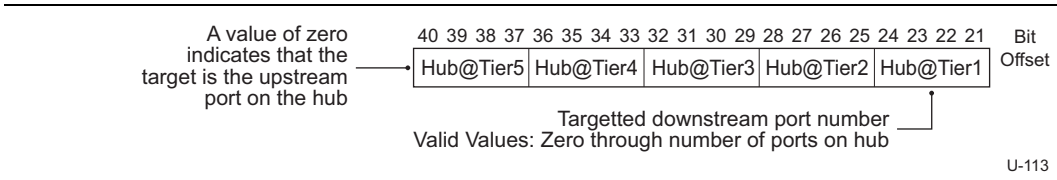


Figure 8-24. Route String Detail

In Figure 8-24 the value in Hub@Tier1 field is the downstream port number of the hub connected directly to one of the root ports on the host to which a second hub is attached and so on.

8.9.1 Route String Port Field

This 4-bit wide field in the **Route String** represents the port in the hub being addressed.

8.9.2 Route String Port Field Width

The **Route String** Port field width is fixed at 4 bits, limiting the maximum number of ports a hub may support to 15.

8.9.3 Port Number

The specific port on a hub to which the packet is directed is identified by the value in the Route String Port field. When addressing the hub controller then the Port Number field at the hub's tier level shall be set to zero in the Route String. The hub's downstream ports are addressed beginning with one and count up sequentially.

8.10 Transaction Packet Usages

TPs are used to report the status of data transactions and can return values indicating successful reception of data packets, command acceptance or rejection, flow control, and halt conditions.

8.10.1 Flow Control Conditions

This section describes the interaction between the host and a device when an endpoint returns a flow control response. The flow control is at an end-to-end level between the host and the endpoint on the device. Only bulk, control and interrupt endpoints may send flow control responses. Isochronous endpoints cannot send flow control responses.

An IN endpoint shall be considered to be in a flow control condition if it returns one of the following responses to an ACK TP:

- Responding with an NRDY TP; note that an endpoint shall wait until it receives an ACK TP for the last DP it transmitted before it can send an NRDY TP
- Sending a DP with the **EOB** field set to 1 in the DPH

An OUT endpoint shall be considered to be in a flow control condition if it returns one of the following responses to a DP:

- Responding with an NRDY TP
- Sending an ACK TP with the **NumP** field set to 0

The **Packets Pending** field is only valid when set by the host and does not affect whether or not an endpoint enters the flow control state. Refer to Section 8.11 for further details on host and device TP responses.

When an endpoint is in a flow control condition, it shall send an ERDY TP to be moved back into the active state. Further, if the endpoint is an IN endpoint, then it shall wait until it receives an ACK TP for the last DP it transmitted before it can send an ERDY TP. When an endpoint is not in a flow control condition, it shall not send an ERDY TP unless the endpoint is a Bulk endpoint that supports streams. Refer to Section 8.12.1.4.2 and Section 8.12.1.4.3 for further information about when a Bulk endpoint that supports streams can send an ERDY TP. Note that the host may resume transactions to any endpoint – even if the endpoint had not returned an ERDY TP after returning a flow control response. To ensure that the host and the device continue to operate normally, a host shall ignore ERDY TPs from an endpoint that is not in a flow control state.

8.10.2 Burst Transactions

The SuperSpeed USB protocol allows the host to continually send data to a device or receive data from a device as long as the device can receive the data or transmit the data. The number of packets an endpoint on a device can send or receive at a time without an intermediate acknowledgement packet is reported by the device in the endpoint companion descriptor (refer to Section 9.6.7) for that endpoint. An endpoint that reports more than one packet in its maximum burst size is considered to be able to support "Burst" Transactions.

While bursting the following rules apply:

- The maximum number of packets that can be sent in a burst prior to receiving an acknowledgement is limited to the minimum of the maximum burst size (see the definition of bMaxBurst in Table 9-20) of the endpoint and the value of the **NumP** field in the last ACK TP or ERDY received by the endpoint or the host, minus the number of packets that the endpoint or the host has already sent after the packet acknowledged by the last ACK TP.

  Note that host may re-initialize the maximum number of DPs that can be sent/received in a burst to the maximum burst size of the endpoint whenever the endpoint is initialized.

- Each individual packet in the burst shall have a data payload of maximum packet size. Only the last packet in a burst may be of a size smaller than the reported maximum packet size. If the last one is smaller, then the same rules for short packets apply to a short packet at the end of a burst (refer to Section 8.10.3).

- The burst transaction continues as long as the **NumP** field in the ACK TP is not set to zero and each packet has a data payload of maximum packet size.

- The **NumP** field can be incremented at any time by the host or a device sending the ACK TP as long as the device or host wants to continue receiving data. The only requirement is that the **NumP** field shall not have a value greater than the maximum burst supported by the device. However, for an ISOC IN endpoint, refer to Section 8.12.6.1 on additional requirements of how to change **NumP** for each burst.

- If a device or host sending an ACK TP decrements the **NumP** field, then it shall do so by no more than one. For example, if the previous ACK TP had a value of five in the **NumP** field, then the next ACK TP to acknowledge the next packet received shall have a value of no less than four in the **NumP** field. The only exceptions to this rule are:
  - If the device can receive the data but cannot accept any more data, then it shall send an ACK TP with the **NumP** field set to zero.
  - The host shall send an ACK TP with the **NumP** field set to zero in response to a device sending a DP with the **EOB** field set or that is a short packet (see Section 8.10.3). However, if the host receives a short packet with EOB = 0 and the host has another transfer to initiate with the same endpoint, then the host may instead send an ACK TP with the NumP field set to a non-zero value.
  - The host may send an ACK TP with the **rty** bit set to one and the **NumP** field set to any value less than the maximum burst that the endpoint is capable of, including zero, in response to a device sending a DP with a DPP error (See Section 8.11.2).

8.10.3 Short Packets

SuperSpeed retains the semantics of short packet behavior that USB 2.0 supports. When the host or a device receives a DP with the **Data Length** field shorter than the maximum packet size for that endpoint it shall deem that that transfer is complete.

In the case of an IN transfer, a device shall stop sending DPs after sending a short DP. The host shall respond to the short DP with an ACK TP with the **NumP** field set to zero unless it has another transfer for the same endpoint in which case it may set the **NumP** field as mentioned in Section 8.10.2. The host shall schedule transactions to the endpoint on the device when another transfer is initiated for that endpoint.

In the case of an OUT transaction, the host may stop sending DPs after sending a short DP. The host shall schedule transactions to an endpoint on the device when another transfer is initiated for that endpoint. Note that this shall be the start of a new burst to the endpoint.

8.11 TP or DP Responses

Transmitting and receiving devices shall return DPs or TPs as detailed in Table 8-25 through Table 8-27. Not all TPs are allowed, depending on the transfer type and depending on the direction of flow of the TP.

8.11.1 Device Response to TP Requesting Data

Table 8-25 shows the possible ways a device shall respond to a TP requesting data for bulk, control, and interrupt endpoints. A TP is considered to be invalid if it has an incorrect Device Address or the endpoint number and direction does not refer to an endpoint that is part of the current configuration or it does not have the expected sequence number.

Table 8-25. Device Responses to TP Requesting Data (Bulk, Control, and Interrupt Endpoints)

Invalid TP Received	TP Received with Deferred Bit Set	Device Tx Endpoint Halt Feature Set	Device Ready to Transmit Data	Action Taken
Yes	Do not care	Do not care	Do not care	The device shall ignore the TP.
No	Yes	Yes	Do not care	The device shall send an ERDY TP.
No	Yes	No	No	The device shall not respond. It shall send an ERDY TP when it is ready to resume.
No	Yes	No	Yes	The device shall send an ERDY TP indicating that it is ready to send data.
No	No	Yes	Do not care	Issue STALL TP
No	No	No	No	Issue NRDY TP
No	No	No	Yes	Start transmitting DPs with sequence numbers requested by the host

Note that an IN endpoint shall wait until it receives an ACK TP for the last DP it transmitted before it can send an STALL TP.

8.11.2 Host Response to Data Received from a Device

Table 8-26 shows the host responses to data received from a device for bulk, control, and interrupt endpoints. The host is able to return only an ACK TP. A DPH is considered to be invalid if it has an incorrect Device Address or the endpoint number and direction does not refer to an endpoint that is part of the current configuration or it does not have the expected sequence number or the Data length in the DPH is greater than the endpoint's maximum packet size. In Table 8-26, DPP Error may be due to one or more of the following:

- CRC incorrect
- DPP aborted
- DPP missing
- Data length in the DPH does not match the actual data payload length

Table 8-26. Host Responses to Data Received from a Device (Bulk, Control, and Interrupt Endpoints)

DPH has Invalid Values	Data Packet Payload Error	Host Can Accept Data	TP Returned by Host
Yes	Do not care	Do not care	Discard data and do not send any TP.
No	Yes	Do not care	Discard data and send an ACK TP with the **Retry** bit set, requesting for zero or more DPs with the **Sequence Number** field set to the sequence number of the DP that was corrupted.
No	No	No	Discard data; send an ACK TP with the **Retry** bit set requesting for one or more DPs with the **Sequence Number** field set to the sequence number of the DP that the host was unable to receive. The ACK TP shall have the **Host Error** bit set to one to indicate that the host was unable to accept the data.
No	No	Yes	Accept data and send an ACK TP requesting for zero or more DPs with the **Sequence Number** field set to the sequence number of the next DP expected. This is also an implicit acknowledgement that this DP was received successfully.

8.11.3 Device Response to Data Received from the Host

TP responses by a device to data received from the host for bulk, control, and interrupt endpoints are shown in Table 8-27. A DPH is considered to be invalid if it has an incorrect Device Address or the endpoint number and direction does not refer to an endpoint that is part of the current configuration or it does not have the expected sequence number or the Data length in the DPH is greater than the endpoint's maximum packet size. In Table 8-27, DPP Error may be due to one or more of the following:

- CRC incorrect
- DPP aborted
- DPP missing
- Data length in the DPH does not match the actual data payload length

Note: Receipt of an ACK TP indicates to the host the DP with the previous sequence number was successfully received by a device as well as the number of data packet buffers the device has available to receive any pending DPs the host has. A device shall send an ACK TP for each DP received.

Table 8-27. Device Responses to OUT Transactions (Bulk, Control, and Interrupt Endpoints)

DPH has Invalid Values	DPH has Deferred Bit Set	Receiver Halt Feature Set	Data Packet Payload Error	Device Can Accept Data	TP Returned by Device
Yes	Do not care	Do not care	Do not care	Do not care	Discard DP.
No	Yes	Yes	Do not care	Do not care	The device shall send an ERDY TP.
No	Yes	No	Do not care	No	The device shall not respond. It shall send an ERDY TP when it is ready to resume.
No	Yes	No	Do not care	Yes	The device shall send an ERDY TP.
No	No	Yes	Do not care	Do not care	The device shall send a STALL TP.
No	No	No	Do not care	No	Discard DP, send an NRDY TP.
No	No	No	Yes	Yes	Discard DP, send an ACK TP with the sequence number of the DP expected (thereby indicating that the DP was not received), the Retry bit set and the number of DPs that the device can receive for this endpoint.
No	No	No	No	Yes	Send an ACK TP indicating the sequence number of the next DP expected (thereby indicating that this DP was received successfully) and the number of DPs that the device can receive for this endpoint.

8.11.4 Device Response to a SETUP DP

A SETUP DP is a special DP that is identified by the **Setup** field set to one and addressed to any control endpoint. SETUP is a special type of host-to-device data transaction that permits the host to initiate a command that the device shall perform. Upon receiving a SETUP DP, a device shall respond as shown in Table 8-28.

Note that a SETUP DPH shall be considered invalid if it has any one of the following:

- Incorrect Device Address
- Endpoint number and direction does not refer to an endpoint that is part of the current configuration
- Endpoint number does not refer to a control endpoint
- Non-zero sequence number
- Data length is not set to eight

In Table 8-28, DPP Error may be due to one or more of the following:

- CRC incorrect
- DPP aborted
- DPP missing
- Data length in the Setup DPH does not match the actual data payload length.

Table 8-28. Device Responses to SETUP Transactions (Only for Control Endpoints)

DPH has Invalid Values	DPH has Deferred Bit Set	Data Packet Payload Error	TP Returned by Device
Yes	N/A	N/A	Discard DP.
No	Yes	N/A	The device shall send an ERDY TP indicating that it is ready to receive the SETUP DP.
No	No	Yes	Discard SETUP DP, send an ACK TP with the sequence number set to zero, the rty bit set and the **NumP** field set to one.
No	No	No	Send an ACK TP with the sequence number set to one (thereby indicating that this SETUP DP was received successfully). The value in the **NumP** field indicates to the host whether the device wants to flow control the Data/Status stage or not. Refer to Section 8.12.2 for details.

8.12 TP Sequences

The packets that comprise a transaction vary depending on the endpoint type. There are four endpoint types: bulk, control, interrupt, and isochronous.

8.12.1 Bulk Transactions

Bulk transaction types are characterized by the ability to guarantee error-free delivery of data between the host and a device by means of error detection and retry. Bulk transactions use a two-phase transaction consisting of TPs and DPs. Under certain flow control and halt conditions, the data phase may be replaced with a TP.

8.12.1.1 State Machine Notation Information

This section shows detailed host and device state machines required to advance the Protocol on an IN or OUT pipe. The diagrams should not be taken as a required implementation, but to specify the required behavior.

Figure 8-25 shows the legend for the state machine diagrams. A circle with a three line border indicates a reference to another (hierarchical) state machine. A circle with a two line border indicates an initial state. A circle with a single line border is a simple state.

A diamond (joint) is used to join several transitions to a common point. A joint allows a single input transition with multiple output transitions or multiple input transitions and a single output transition. All conditions on the transitions of a path involving a joint must be true for the path to be taken. A path is simply a sequence of transitions involving one or more joints.

A transition is labeled with a block with a line in the middle separating the (upper) condition and the (lower) actions. A transition without a line is a condition only. The condition is required to be true to take the transition. The actions are performed if the transition is taken. The syntax for actions and conditions is VHDL. A circle includes a name in bold and optionally one or more actions that are performed upon entry to the state.

Transitions using a solid arrow are generated by the host. Transitions using a dashed arrow are generated by a device. Transitions using a dot-dot-dash arrow are generated by the either a device or the host.

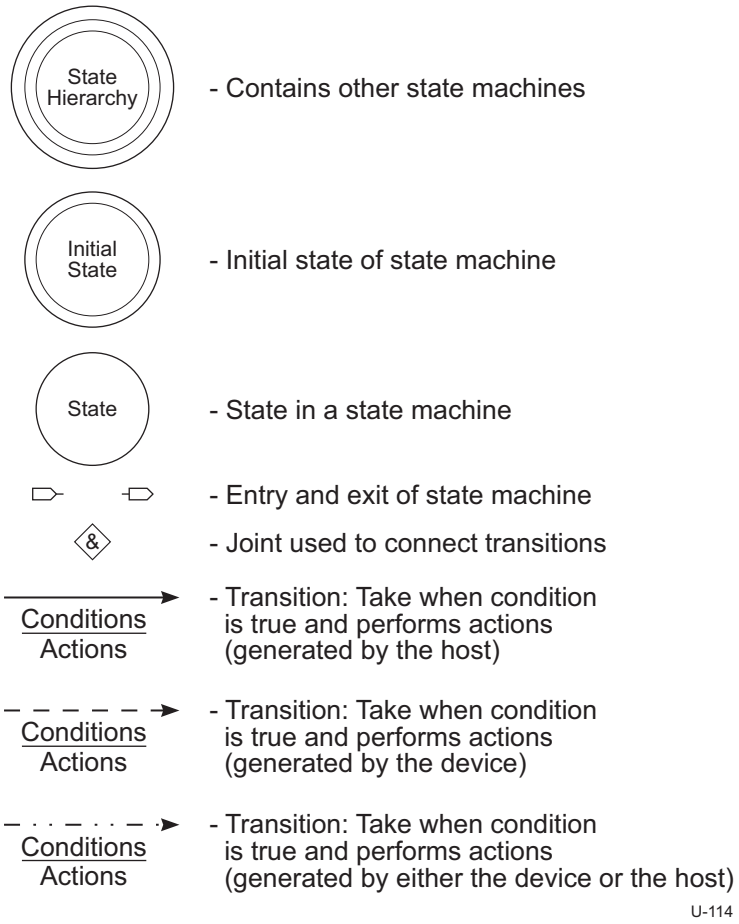


Figure 8-25. Legend for State Machines

8.12.1.2 Bulk IN Transactions

When the host is ready to receive bulk data, it sends an ACK TP to a device indicating the sequence number and number of packets it expects from the device. A Bulk endpoint shall respond as defined in Section 8.11.1.

The host shall send an ACK TP for each valid DP it receives from a device. A device does not need to wait for the ACK TP to send the next DP to the host if the previous ACK TP indicated that the host expected the device to send more than one DP (depending on the value of the **Number of Packets** field in the TP). The ACK TP implicitly acknowledges the last DP with the previous sequence number as being successfully received by the host and also indicates to the device the next DP with the sequence number and number of packets the host expects from the device. If the host detects an error while receiving any of the DPs, it shall send an ACK TP with the sequence number value set to the first DP that was received with an error with the Retry bit set, even if subsequent packets in the burst asked for by the host were received without error. A device is required to resend all DPs starting from the sequence number set in the ACK TP in which the Retry bit set.

The host expects the first DP to have a sequence number set to zero when it starts the first transfer from an endpoint after the endpoint has been initialized (via a Set Configuration, Set Interface, or a ClearFeature (ENDPOINT_HALT) command – refer to Chapter 9 for details on these commands). The second DP sent by the device from that endpoint shall have a sequence number set to one; the third DP has a sequence number set to two; and so on until sequence number 31. The next DP after sequence number 31 uses a sequence number of zero. An endpoint on the device keeps incrementing the sequence number of the packets it transmits unless it receives an ACK TP with the Retry bit set to one that indicates that it has to retransmit an earlier DP.

If the host asks for multiple DPs from a device and the device does not have that number of DPs available to send at the time, the device shall send the last DP with the **End Of Burst** flag in the DPH set to one. Note that it is not necessary to set the **End Of Burst** flag if the DP sent to the host has a payload that is less than the MaxPacketSize for that endpoint.

A transfer is complete when a device sends all the data that is expected by the host or it sends a DP with a payload that is less than the MaxPacketSize. When the host wants to start a new transfer, it shall send another ACK TP with the next sequence number and number of DPs expected from a device. For example, if the DP with the payload less than MaxPacketSize was two, the host shall initiate the next transfer by sending an ACK TP with the expected sequence number set to three.

8.12.1.3 Bulk OUT Transactions

When the host is ready to transmit bulk data, it sends one or more DPs to a device. If a DPH with valid values (valid device address, endpoint number, and direction as well as the expected sequence number) is received by a device, it shall respond as defined in Section 8.11.3.

The host always initializes the first DP sequence number to zero in the first transfer it performs to an endpoint after the endpoint is initialized (via a Set Configuration, Set Interface, or ClearFeature (ENDPOINT_HALT) command – refer to Chapter 9 for details on these commands). The second DP has a sequence number set to one; the third DP has a sequence number set to two; and so on until 31. The next DP after sequence number 31 uses a sequence number of zero. The host keeps incrementing the sequence number of the DPs it transmits unless it receives an ACK TP with the Retry bit set to one that indicates that it has to retransmit an earlier packet.

A transfer is complete when the host sends all the data it has to a device; however the last DP of the transfer may or may not have a payload which is equal to the MaxPacketSize of the endpoint. When the host wants to start a new transfer it shall send another DP, with the next sequence number, targeted at an endpoint in the device.

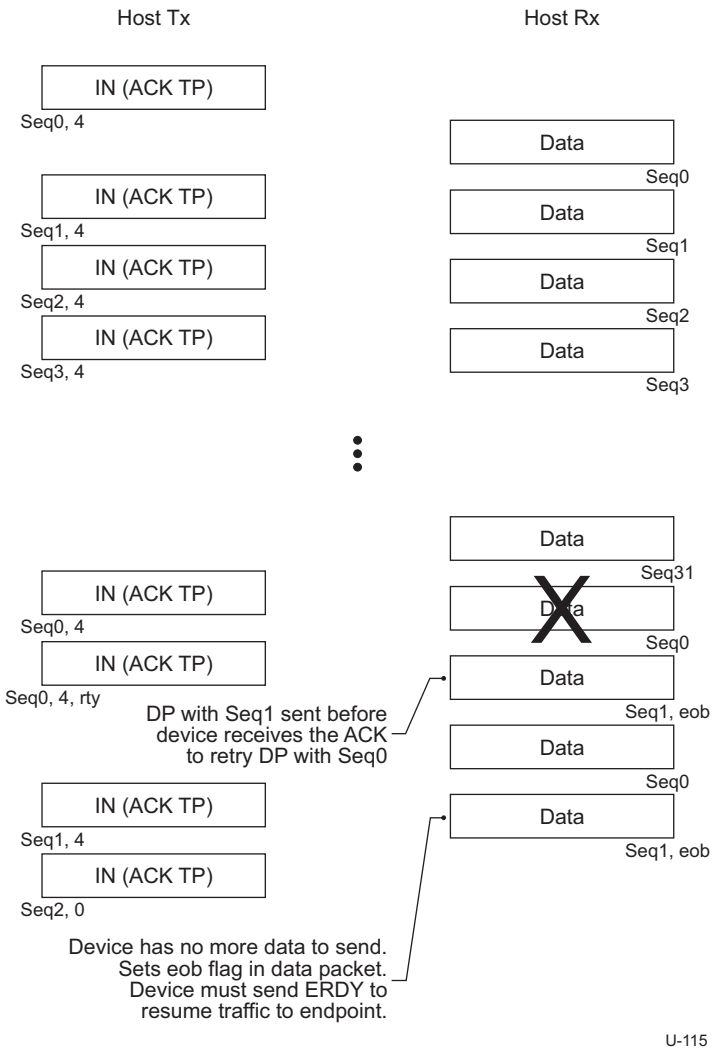


Figure 8-26. Sample BULK IN Sequence

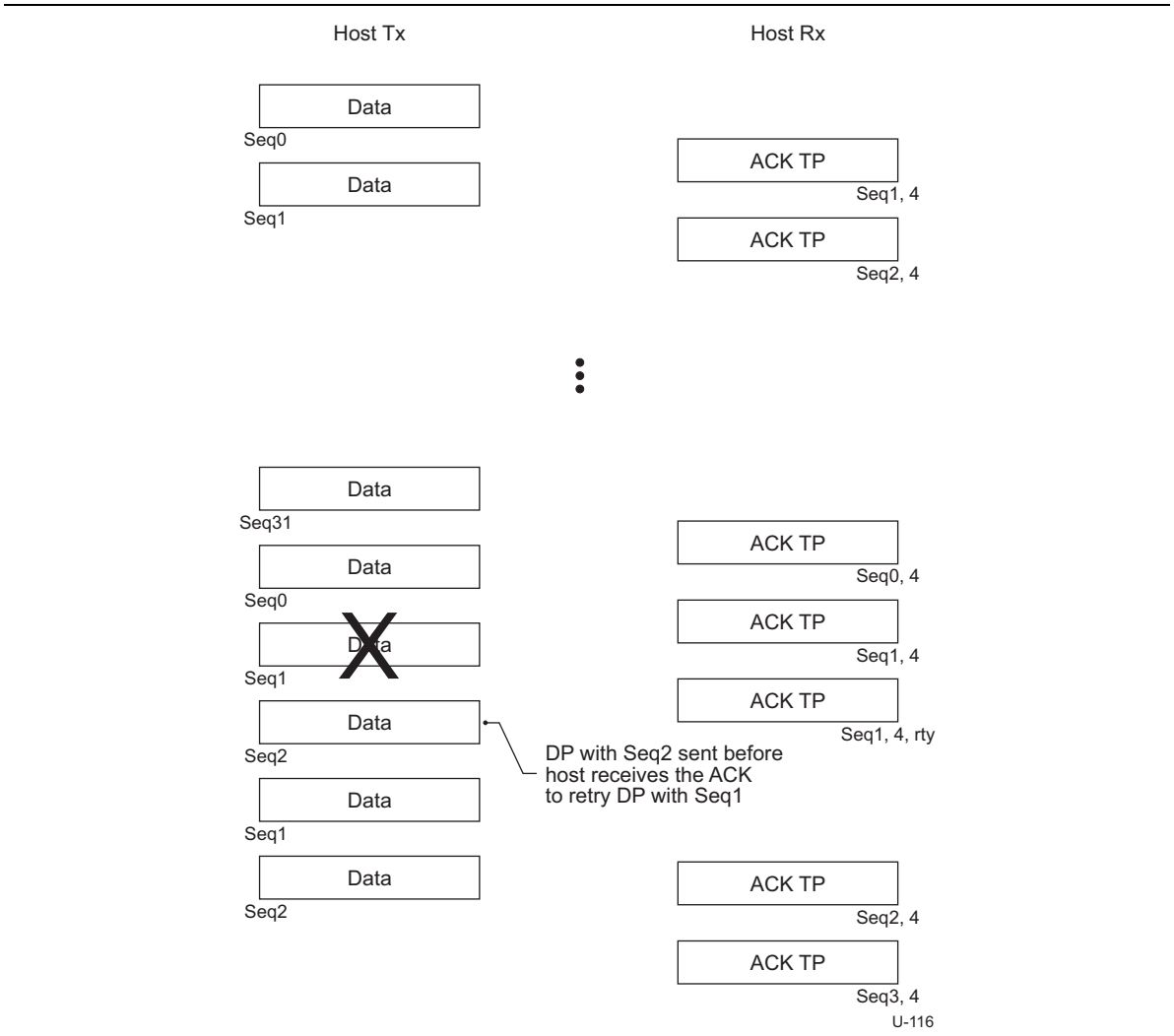


Figure 8-27. Sample BULK OUT Sequence

8.12.1.4 Bulk Streaming Protocol

The Stream Protocol adheres to the semantics of the standard SuperSpeed Bulk protocol, so the packet exchanges on a SuperSpeed bulk pipe that supports Streams are indistinguishable from a SuperSpeed bulk pipe that does not. The Stream Protocol is managed strictly through manipulation of the *Stream ID* field in the packet header.

Note: Device Class defined methods are used for coordinating the Stream IDs that are used by the host to select Endpoint Buffers and by the device to select the Function Data associated with a particular Stream. Typically this is done via an out-of-band mechanism (e.g., another endpoint) that is used to pass the list of "Active Stream IDs" between the host and the device.

Note: The Stream state machines illustrate a 1:1 relationship between sending a DP and receiving an ACK. Logically this is true, however SuperSpeed burst capabilities allows up to MaxBurst outstanding ACKs between the host and a device so temporally there may be a "many to 1"

relationship. Bursts are managed on a Stream pipe identically to how they are managed on a normal Bulk pipe. Refer to Section 8.10.2 for more information on Burst Transactions.

Note: As described in this section, the Stream Protocol applies to the state of the “pipe” and is described as single entity. In reality, the Stream Protocol is being tracked independently by the host at one end of the pipe and the device at the other. So at any instant in time the two ends may momentarily be out of phase due to packet propagation delays between the host and the device.

Note: If a Retry is requested and the host cannot continue retransmission of a DP during the current burst, the host shall return to the endpoint at the next available opportunity within the constraints of the transfer type.

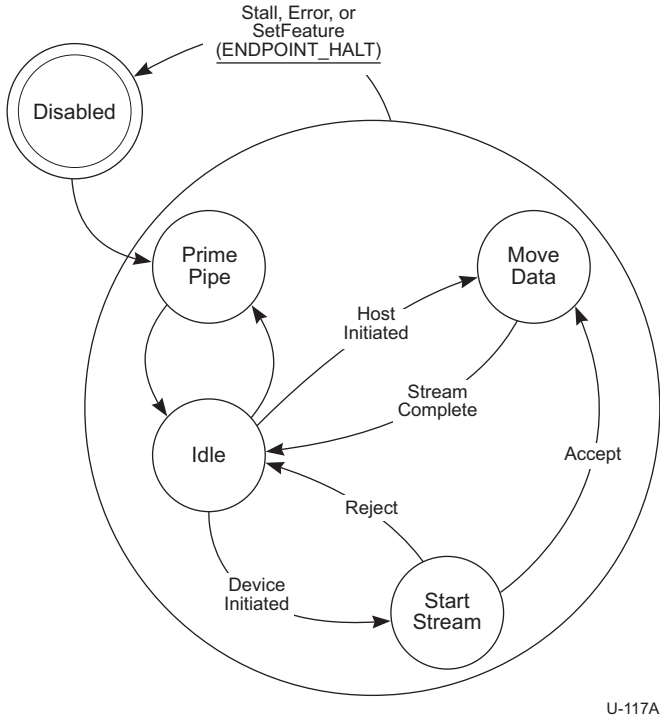


Figure 8-28. General Stream Protocol State Machine (SPSM)

Figure 8-28 illustrates the basic state transitions of the Stream Protocol State Machine (SPSM). This section describes the general transitions of the SPSM as they apply to both IN and OUT endpoints. Detailed operation of the SPSM for IN and OUT endpoints is described in subsequent sections.

Disabled – This is the initial state of the pipe after it is configured, as well as the state that is transitioned to if an error is detected in any of the other states. The first time an Endpoint Buffer is assigned to the pipe, the host shall transition the SPSM to the Prime Pipe state. If the **Disabled** state was entered due to an error, then the error condition must be removed by software intervention before the state may be exited.

Note that an error (e.g., Stall) or a SetFeature(ENDPOINT_HALT) request shall transition any SPSM state to the **Disabled** state. If an error condition is detected by the host (e.g., Stall,

tHostTransactionTimeouts, etc.), the host shall transition its SPSM for the endpoint to the Disabled state. In the case where the host detects an error, not asserted by device (e.g., tHostTransactionTimeouts), the host shall transition the device's SPSM to the Disabled state by issuing a SetFeature(ENDPOINT_HALT) request to the device.

Prime Pipe – A transition to this state is always initiated by host, and informs a device that an Endpoint Buffer set has been added or modified by software. After exiting this state, any Active Stream IDs previously considered Not Ready by the device shall now be considered Ready.

Note: To minimize bus transactions, the host controller limits transitions to the Prime Pipe state to one transition per Idle state entry. This means that while in the Idle state only a single transition to the Prime Pipe state will be generated even if Endpoint Buffers for multiple streams become ready. And since the Prime Pipe state does not specify which Stream(s) are ready, all Active Stream IDs are set to Ready by a Prime Pipe. The device is responsible for testing all Active Stream IDs (as described above) by sending the appropriate ERDYs after returning to Idle. Note that Device Class defined constraints may be used to limit the number of Active Stream IDs that need to be tested at any point in time.

Idle – A transition to this state indicates that there is no Current Stream (CStream) selected. In this state, the SPSM is waiting for a transition to Prime Pipe or Move Data initiated by the Host, or a transition to Start Stream initiated by the Device. The object of the Host and Device Initiated transitions is to start moving data for a Stream. A host initiated transition to Move Data is referred to as a Host Initiated Move Data or HIMD. All Active Stream IDs are set to Ready by a HIMD.

Start Stream – This state is always initiated by a Device, and informs the host that the device wants to begin moving data on a selected Stream. The device may initiate a transition to this state anytime it has a Ready Stream ID. If the device selected Stream is accepted by the host, then the pipe enters the Move Data state. If the device selected Stream ID is rejected by the host, the pipe returns to Idle state and the selected Stream ID shall temporarily be considered Not Ready by the device. Note that a device maintains a list of the "Active" Stream IDs. An Active Stream ID may be Ready or Not Ready. The device is informed of the Active Stream IDs by the host through an out-of-band mechanism (typically a separate OUT endpoint).

Move Data – In this state, Stream data is transferred. The Current Stream is set when the SPSM transitions to this state. The SPSM transitions to the Idle state when the Stream transfer is complete, or if the host or device decides to terminate the Stream transfer because they have temporarily exhausted their data or buffer space. The transition to Idle invalidates the Current Stream for the pipe.

Note: The general rule is that a Stream state machine advances only due to the reception of a good DP or TP. For example, if a DP is received with a bad DPP, a Stream state machine shall perform any retries in the current state, and advance only if a good packet is transferred.

8.12.1.4.1 Stream IDs

A 16-bit field *Stream ID* field is reserved in DP headers and in ACK, NRDY, and ERDY TPs for passing SIDs between the host and a device. Specific SID values that are reserved by the Stream Protocol and other SID notations are:

- **NoStream** – This SID indicates that no Stream ID is associated with the respective bus packet and the Stream ID field should not be interpreted as referencing a valid Stream. The *NoStream* SID value is FFFFh.

- **Prime** – This SID is used to define transitions into and out of the Prime Pipe state. As with *NoStream*, no Stream ID is associated with the respective bus packet and the Stream ID field should not be interpreted as referencing a valid Stream. The *Prime* SID value is FFFEh.
- **Stream n** – Where n is a value between 1 and 65533 (FFFDh). This notation is used to reference a valid Stream ID. The Stream ID field in the packet header is valid if it uses this notation. Valid *Stream n* SID values are between 1 and 65533 (FFFDh), where the numeric value is identical to *n*.
- **Stream 0** – This value is reserved and not used by a pipe that supports Streams. The *Stream 0* SID value is 0000h. Its use is required by a standard bulk pipe.
- **CStream** – represents the value of the "Current" Stream ID assigned to the pipe. A *CStream* value is maintained by both the host and a device. The Stream Protocol ensures that the *CStream* values are consistent in the host and the device. Valid values are *NoStream* or *Stream n*.
- **LCStream** – represents the value of the CStream SID assigned to the pipe before the last state transition. An *LCStream* value is maintained by the host. Valid values are *Prime*, *NoStream*, or *Stream n*. For example, while the pipe in the Move Data state CStream = Stream n, when the pipe transitions from Move Data to Idle state, LCStream is set to *Stream n*, and CStream is set to *NoStream*, thus LCStream records the "Last CStream" value.

Stream n SID values are assigned by the host and passed to a device (typically through an out-of-band, Device Class defined method). The value of a *Stream n* SID shall be treated as a "logical value" by a device, i.e., the device should not infer any meaning from the value or modify it.

Note: The Bulk IN and OUT Stream Protocols below describe simplified state machines that do not explicitly detail the burst feature of SuperSpeed endpoints which allows DPs to be sent without receiving an ACK. An implementation shall extend these state machines to manage bursting.

The following sections (8.12.1.4.2 to 8.12.1.4.5) separate the Stream state machines into four cases for the device and host ends of a Stream pipe. Sections 8.12.1.4.2 and 8.12.1.4.3 describe the device end state machines. Sections 8.12.1.4.4 and 8.12.1.4.5 describe the host end state machines. And for each end of the pipe a separate section describes the respective IN and OUT operations.

The subsections in each Stream state machine section describe the state machine's respective states. The subsections begin with a description of the purpose and general characteristics of the state, followed by a discussion of each of the state's exit transitions. A paragraph that describes a state's exit transition is preceded with a unique *condition* or *action* label of the associated exit transition in the previous state diagram figure.

Note: The U1 or U2 Timeouts in the path between the host and a device should be set to values that will prevent a transition to a U1 or U2 state for normal responses to Data Transactions. Refer to Section 8.13 for more Data Transaction timing information.

Note: In the Stream state machine sections, the state names are overloaded, e.g., The **Idle** state is defined in all four state machine descriptions. The **INMvData Host** state is defined in both the device and host IN state machine sections, etc. The states are related in that they may occur at either end of a Stream pipe, however each Stream state machine section describes an independent state machine, so the conditions and actions associated with the states are distinct in each section.

Note: A transition condition that is italicized shall be interpreted as a comment, not a required condition. For example, the "*Stream n Active and Ready*" text of the **Idle** to **Start Stream** transition of Figure 8-29.

Note: Any *CStream* data payload may be zero-length. The use of zero-length DPs on a Stream pipe (other than for Prime Pipe or Start Stream reject operations) is defined by the Device Class associated with the endpoint.

Note: An IN Data or Burst Transaction is terminated with an ACK TP with NumP = 0. This ACK TP is referred to as a "Terminating ACK" in the following sections.

8.12.1.4.2 Device IN Stream Protocol

This section defines the SuperSpeed packet exchanges that transition the device side of the Stream Protocol from one state to another on an IN bulk endpoint.

In the following text, a Device IN Stream state transition is assumed to occur at the point the device sends the first bit of the first symbol of a state machine related message to the host, or at the point the device first decodes state machine related message from the host.

For an IN pipe, Endpoint Buffers in the host receive Function Data from a device.

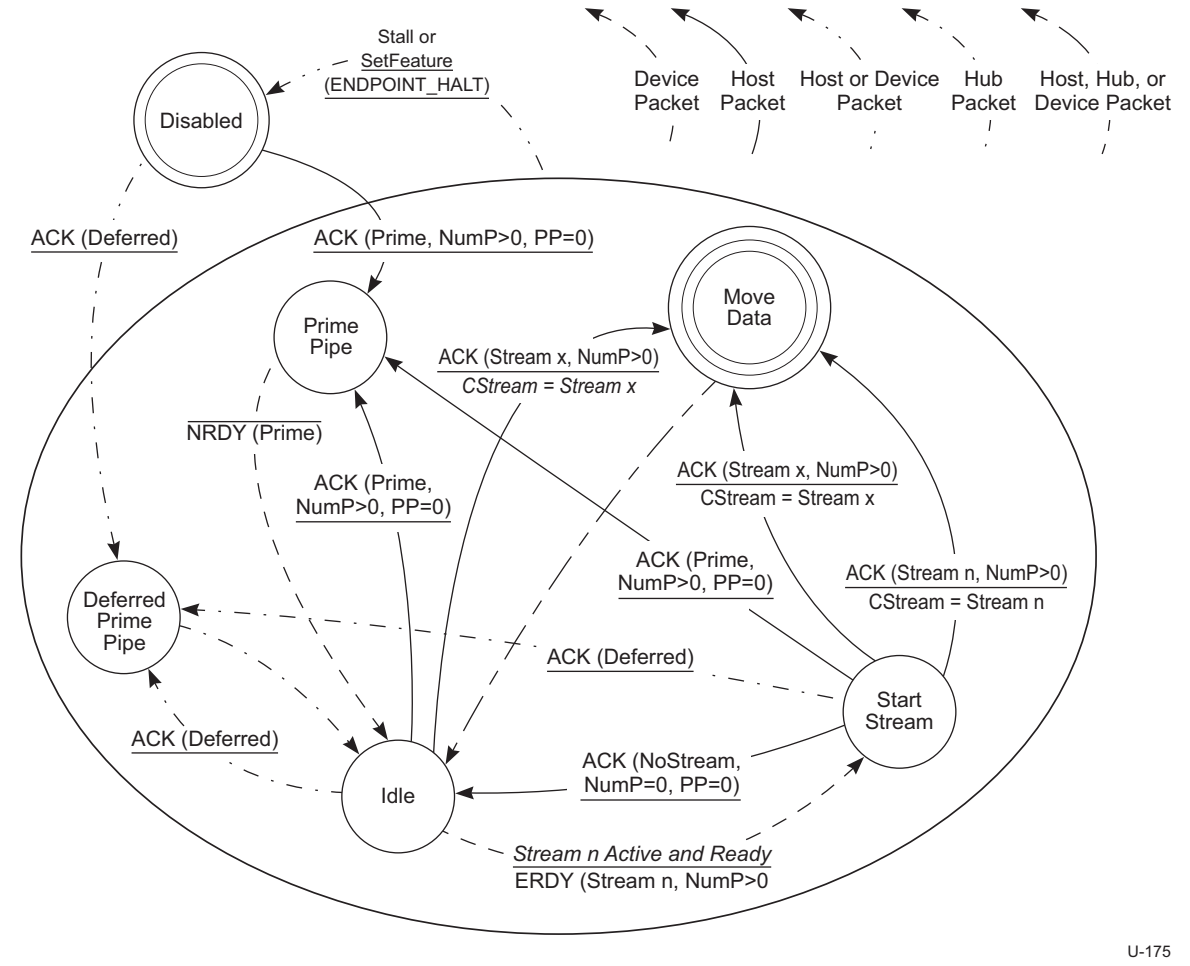


Figure 8-29. Device IN Stream Protocol State Machine (DISPSM)

8.12.1.4.2.1 Disabled

After an endpoint is configured or receives a SetFeature(ENDPOINT_HALT) request, the pipe is in the **Disabled** state.

ACK(Prime, NumP>0, PP=0) - If an ACK TP with the Stream ID field set to *Prime* is received, then the device shall transition the pipe to the **Prime Pipe** state. This transition occurs after the initial Endpoint Buffers are assigned to the pipe by system software.

ACK(Deferred) - If an ACK with the Deferred (DF) flag set is received, then the device shall transition the pipe to the **Deferred Prime Pipe** state. This packet is received when the link has transitioned to a U1 or U2 state while waiting for the initial Endpoint Buffer assignment.

8.12.1.4.2.2 Prime Pipe

The **Prime Pipe** state informs the device that the Endpoint Buffers have been assigned to one or more Streams, however it does not specify which Stream(s). In this state, the device shall set all Active Streams to Ready. After returning to the **Idle** state the device shall issue an ERDY to start a specific Stream from its list of Active Streams.

NRDY(Prime) - Upon entering the **Prime Pipe** state, the device shall generate an NRDY TP with its Stream ID field set to *Prime* and transition to the **Idle** state.

8.12.1.4.2.3 Deferred Prime Pipe

The **Deferred Prime Pipe** state informs the device that the Endpoint Buffers have been assigned to one or more Streams, however the link has transitioned to a U1 or U2 state while waiting. In this state, the device shall set all Active Streams to Ready. After returning to the **Idle** state the device shall issue an ERDY to start a specific Stream from its list of Active Streams.

No Condition - Upon entering the **Deferred Prime Pipe** state, the device shall immediately transition to the **Idle** state. This is the only **Deferred Prime Pipe** exit transition in Figure 8-29.

8.12.1.4.2.4 Idle

In the **Idle** state, the pipe is waiting for a Stream selection (e.g., a transition to **Start Stream** or **Move Data**) or a notification from the host that a Stream Endpoint Buffer has been added or modified for the pipe (i.e., transition to **Prime Pipe**). Note that upon the initial entry in to **Idle** (i.e., from **Disabled**), only the device may initiate a Stream selection.

ERDY(Stream n, NumP>0) - To initiate a Stream selection, the device generates an ERDY TP with its Stream ID set to *Stream n* and a NumP value > 0, and transitions to the **Start Stream** state, where *Stream n* is the Stream ID proposed by the device. A device may initiate this transition when it wishes to start a Stream transfer, regardless of whether the pipe is in a flow control condition or not. The device maintains a list of *Active and Ready* Streams that it may generate ERDYs for. The method that a device uses for Stream selection is outside the scope of this specification and is normally defined by the Device Class associated with the pipe. Note that the value of the ERDY NumP field reflects the amount of Endpoint Data the device has available for *Stream n*.

ACK(Prime, NumP>0, PP=0) - If an ACK TP with a Stream ID equal to *Prime* is received from the host, the device shall transition to the **Prime Pipe** state.

ACK(Stream x, NumP>0) - With this transition the host proposes the Stream ID *Stream x* to the device. If an ACK TP with a Stream ID not equal to *Prime* is received from the host, the device shall transition to the **Move Data** state. The host may initiate this transition when it wishes to start a Stream transfer and is referred to as a *Host Initiated Move Data* or **HIMD**. A HIMD indicates the specific Stream that the Endpoint Buffer had been changed for. The device shall set *Stream x* to Ready due to this transition. After entering the **Move Data** state, the device may reject the proposed Stream with an NRDY or accept the proposed Stream with a DP. Upon transitioning to the **Move Data** state the device sets *CStream* to the value of the received Stream ID (*Stream x*). Typically *Stream x* will be equal to the Stream ID (*Stream n*) in the last ERDY generated by the device. *Stream x* may not be equal to *Stream n* if one of the race conditions described below occurs, because the host drops ERDYs under these conditions. PP should equal 1.

ACK(Deferred) - If an ACK with the Deferred (DF) flag set is received, then the device shall transition the pipe to the **Deferred Prime Pipe** state. This packet is received when the link has transitioned to a U1 or U2 state and the host has attempted a HIMD.

8.12.1.4.2.5 Start Stream

In the **Start Stream** state, the device is waiting for the host to accept or reject the Active and Ready Stream selection that it has proposed.

ACK(Stream n, NumP>0) - If an ACK TP with a Stream ID equal to *Stream n* is received, the host has accepted the device's proposal for starting *Stream n* and the device shall transition to the **Move Data** state. Upon transitioning to the **Move Data** state the device sets *CStream* to the value of the received Stream ID (*Stream n*). PP should equal 1.

ACK(NoStream, NumP=0, PP=0) - If an ACK TP with a Stream ID equal to *NoStream* is received, the host has rejected the device's proposal for starting *Stream n* and the device shall transition to the **Idle** state. The device shall set *Stream n* to Not Ready due to this transition. The host shall reject a proposal from a device if there are no Endpoint Buffers available for it.

ACK(Prime, NumP>0, PP=0) - If an ACK TP with a Stream ID equal to *Prime* is received, a race condition has occurred. The host has entered the **Prime Pipe** state to inform the device that the Endpoint Buffers for one or more Streams have been updated, at the same time that the device has attempted to initiate a Stream transfer, and their respective messages have passed each other on the link. During this condition, the device is in the **Start Stream** state and the host is in the **Prime Pipe** state. To resolve this condition, the device shall transition to the **Prime Pipe** state.

ACK(Stream x, NumP>0) - If an ACK TP with a Stream ID equal to *Stream x* is received, a race condition has occurred. The host has entered the **Move Data** state to initiate a transfer on *Stream x*, at the same time that the device has attempted to initiate a transfer on *Stream n*, and their respective messages have passed each other on the link. During this condition, the device is in the **Start Stream** state and the host is in the **Move Data** state. To resolve this condition, the device shall transition to the **Move Data** state. The device shall set *Stream x* to Ready due to this transition. Upon transitioning to the **Move Data** state the device sets *CStream* to the value of the received Stream ID (*Stream x*). PP should equal 1.

ACK(Deferred) - If an ACK with the Deferred (DF) flag set is received, then the device shall transition the pipe to the **Deferred Prime Pipe** state. This packet is received when the link has transitioned to a U1 or U2 state while waiting for a host response to the Start Stream request. Note that this transition can occur only if the tERDYTimeout has been exceeded.

Note: The statement "PP should equal 1" in the **Idle** and **Start Stream** states, does not require the device to verify that PP equals 1 for the respective transition, however if a device does check the condition it should halt the EP if PP is not equal to 1.

8.12.1.4.2.6 Move Data

In the Device IN **Move Data** state, *CStream* is set to the same value at both ends of the pipe and the pipe may actively move data. The details of the bus transactions executed in the **Move Data** state and its exit conditions are defined in the Device IN Move Data State Machine defined below.

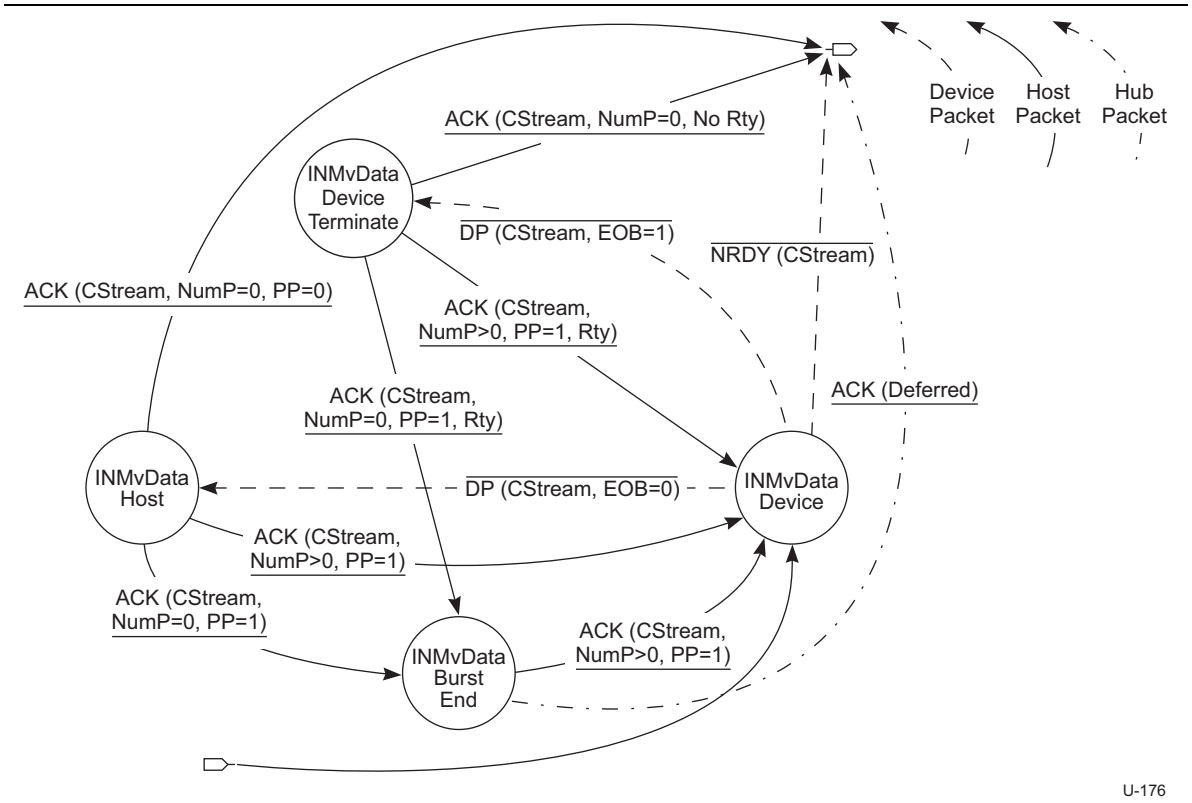


Figure 8-30. Device IN Move Data State Machine (DIMDSM)

The Device IN Move Data State Machine (DIMDSM) is entered from the **Start Stream** or **Idle** states as described above. The entry into the DIMDSM immediately transitions to the **INMvData Device** state. The DIMDSM allows either the device to terminate the Move Data operation because it has exhausted its Function Data associated with a Stream or the host to terminate the Move Data operation because it has exhausted its Endpoint Buffer space associated with a Stream.

The DIMDSM always exits to the **Idle** state. The Retry (Rty=1) flag shall never be set in a packet that causes a DIMDSM exit. A Stream pipe remains in the **Move Data** state during packet retries.

Note: The *Stream ID* value shall be *CStream* for all packets exchanged in the **Move Data** state. If a *Stream ID* value other than *CStream* is detected while in the DIMDSM, the device should halt the endpoint.

Note: if *CStream* is not Active upon initially entering the **Move Data** state, the device may reject the Stream proposal with an NRDY or STALL the pipe, as defined by the associated Device Class.

8.12.1.4.2.7 INMvData Device

This state is initially entered from the **Start Stream** state or the **Idle** state. In this state the device prepares a DP to send to the host or may reject a HIMD from the host.

DP(CStream, EOB=0) - If the device's Endpoint Data for *CStream* is greater than one Max Packet Size, then the device may send a DP to the host with EOB = 0 and transition to the **INMvData Host** state. The DPP shall contain *CStream* data.

DP(CStream, EOB=1) - If the device's Endpoint Data for *CStream* is less than or equal to one Max Packet Size, then the device may send a DP to the host with EOB = 1 and transition to the **INMvData Device Terminate** state. The DPP shall contain *CStream* data.

NRDY(CStream) - The device may reject further *CStream* transfers by sending an NRDY with its Stream ID set to *CStream* and transition to the **Idle** state, exiting the DIMDSM. The device may generate this transition upon initial entry into the DIMDSM to reject a HIMD, or during a Stream transfer due to unexpected internal conditions where it wants to flow control *CStream*.

8.12.1.4.2.8 INMvData Host

In this state the device has just sent a DP to the host and has more Function Data available for *CStream*. The device waits in this state for an acknowledgement from the host for the last DP that it sent.

ACK(CStream, NumP>0, PP=1) - If the device receives an ACK with NumP > 0 and PP = 1, then it shall transition to the **INMvData Device** state. This is the host response if the current burst is not complete and it has more Endpoint Buffer space available for a *CStream* DP from the device. Note that the Retry (Rty=1) flag may be set in this packet if the host detected an error in the last DP from the device. If Rty is set, then the device shall return the DP with the appropriate Sequence Number the next time it sends a DP. If a DP error is detected, the host may continue the current burst until all retries are exhausted or a good DP is received. If the host cannot continue the current burst, the host shall initiate another burst to this endpoint at the next available opportunity within the constraints of the transfer type.

ACK(CStream, NumP=0, PP=1) - If the device receives an ACK with NumP = 0 and PP = 1, then it shall transition to the **INMvData Burst End** state. This is the host response if it has more Endpoint Buffer space available for another *CStream* DP, however it must terminate the current burst from the device. Note that during the **INMvData Host** to **INMvData Device** transitions, the device should see NumP decrement towards 0 as the burst reaches completion. Note that the Retry (Rty=1) flag may be set in this packet if the host detected an error in the last DP from the device.

ACK(CStream, NumP=0, PP=0) - If the device receives an ACK with NumP = 0, and PP = 0, then it shall transition to the **Idle** state, exiting the DIMDSM. This is the host response to a DP when it has accepted the last DP because it has exhausted its *CStream* Endpoint Buffer space. The device shall set *CStream* to Not Ready due to this transition. During the **INMvData Host** to **INMvData Device** transitions, the device should see NumP decrement towards 0 as the Endpoint Buffer is exhausted.

Note: Receiving an ACK with NumP > 0 and PP = 0 is an illegal combination in the **INMvData Host** state and the device should halt the EP if detected.

8.12.1.4.2.9 INMvData Device Terminate

This state is entered because the device has just sent the last DP that it has available for *CStream*, e.g., it has exhausted its *CStream* Function Data. In this state the device waits for an acknowledgement from the host for the last DP of the **Move Data** transfer.

ACK(CStream, NumP=0, No Rty) - If the device receives an ACK with NumP = 0 and Rty = 0, then it shall transition to the **Idle** state, exiting the DIMDSM. This is the normal host response (Terminating ACK) for acknowledging the successful reception of the last DP for *CStream* from the device.

ACK(CStream, NumP>0, PP=1, Rty) - If the device receives an ACK with Rty = 1, then it shall transition to the **INMvData Device** state and resend the appropriate DP. This is the host response if an error was detected on the DP from the device and the burst was not complete.

ACK(CStream, NumP=0, PP=1, Rty) - If the device receives an ACK with Rty = 1 and NumP = 0, then it shall transition to the **INMvData Burst End** state and wait for the host to initiate the next burst. This is the host response if an error was detected on the DP from the device but the burst was complete. The host shall continue the retry process in the next burst.

8.12.1.4.2.10 INMvData Burst End

This state is entered because the host has terminated a burst on a stream pipe. In this state the device waits for an ACK TP that signifies the start of another burst.

ACK(CStream, NumP>0, PP=1) - If the device receives an ACK with NumP > 0 and PP = 1, then it shall transition to the **INMvData Device** state. Note, if the Rty flag was set when the state was entered, then it shall be set upon exit.

ACK(Deferred) - If an ACK with the Deferred (DF) flag set is received, then the device shall transition to the **Idle** state, exiting the DIMDSM. This transition occurs when the link has entered to a U1 or U2 state while waiting for the host to restart a burst, and this transition becomes more likely as the transfer activity associated with other devices increases.

8.12.1.4.3 Device OUT Stream Protocol

This section defines the SuperSpeed packet exchanges that transition the device side of the Stream Protocol from one state to another on an OUT bulk endpoint.

In the following text, a Device OUT Stream state transition is assumed to occur at the point the device sends the first bit of the first symbol of a state machine related message to the host, or at the point the device first decodes a state machine related message from the host.

For an OUT pipe, Endpoint Data in the host is transmitted to Function Buffers in a device. Unless otherwise stated, a DP will contain Endpoint Data.

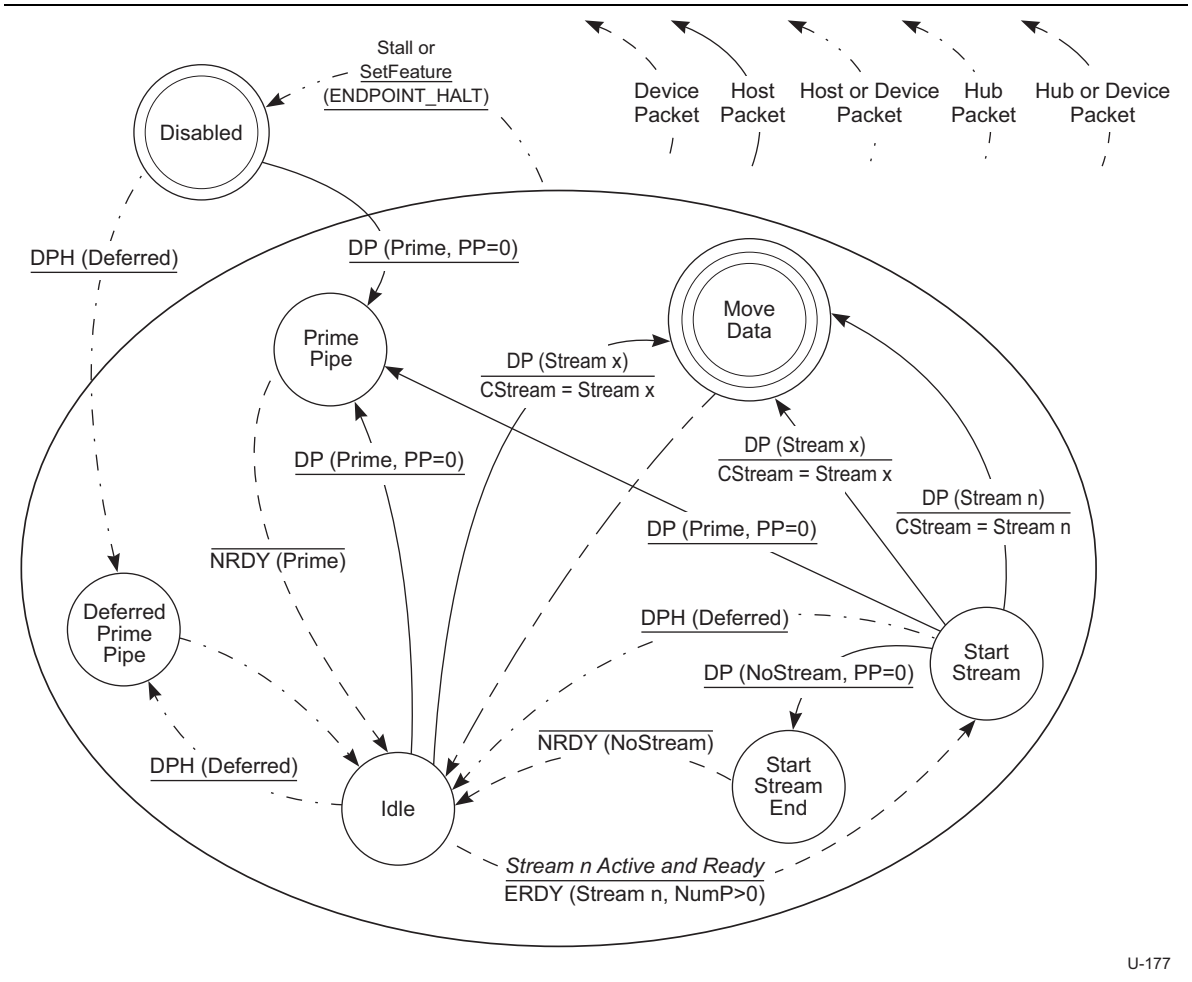


Figure 8-31. Device OUT Stream Protocol State Machine (DOSPSM)

8.12.1.4.3.1 Disabled

After an endpoint is configured or receives a SetFeature(ENDPOINT_HALT) request, the pipe is in the **Disabled** state.

DP(Prime, PP=0) - If a DP with the Stream ID field set to *Prime* is successfully received, then the device shall transition the pipe to the **Prime Pipe** state. The DPP shall contain a zero-length data

payload. This transition occurs after the initial Endpoint Buffers are assigned to the pipe by system software. Note, if an error is detected in the DP data (even though it is zero-length) the device shall remain in the **Disabled** state, and issue ACK(Prime, NumP>0, Rty) packets, retrying until a DP(Prime) is successfully received. This case is not illustrated in the figure above.

DPH(Deferred) - If a DP with the Deferred (DF) flag set is received, then the device shall transition the pipe to the **Deferred Prime Pipe** state. This packet is received when the link has transitioned to a U1 or U2 state while waiting for the initial Endpoint Data assignment.

8.12.1.4.3.2 Prime Pipe

The **Prime Pipe** state informs the device that the Endpoint Data has been assigned to one or more Streams, however it does not specify which Stream(s). After returning to the **Idle** state the device shall issue an ERDY to start a specific Stream from its list of Active and Ready Streams.

NRDY(Prime) - Upon entering the **Prime Pipe** state, the device shall return an NRDY TP with its Stream ID field set to *Prime* and immediately transition to the **Idle** state.

8.12.1.4.3.3 Deferred Prime Pipe

The **Deferred Prime Pipe** state informs the device that the Endpoint Data has been assigned to one or more Streams, however the link has transitioned to a U1 or U2 state while waiting.

No Condition - Upon entering the **Deferred Prime Pipe** state, the device shall immediately transition to the **Idle** state.

8.12.1.4.3.4 Idle

In the **Idle** state, the pipe is waiting for a Stream selection (e.g., a transition to **Start Stream** or **Move Data**) or a notification from the host that Endpoint Data has been added or modified for the pipe (i.e., transition to **Prime Pipe**). Note that upon the initial entry in to **Idle**, only the device may initiate a Stream selection.

ERDY(Stream n, NumP>0) - To initiate a Stream selection, the device generates an ERDY TP with its Stream ID set to *Stream n* and a NumP value > 0, and transitions to the **Start Stream** state, where *Stream n* is the Stream ID proposed by the device. A device may initiate this transition when it wishes to start a Stream transfer, regardless of whether the pipe is in a flow control condition or not. The device maintains a list of *Active and Ready* Streams that it may generate ERDYs for. The method that a device uses for Stream selection is outside the scope of this specification and is normally defined by the Device Class associated with the pipe. Note that the value of ERDY NumP reflects the amount of Endpoint Buffer space the device has available for *Stream n*.

DP(Prime, PP=0) - If a DP with a Stream ID equal to *Prime* is successfully received, the device shall transition to the **Prime Pipe** state. The DPP shall contain a zero-length data payload. Note, if an error is detected in the DP data the device shall remain in the **Idle** state, and issue ACK(Prime, NumP>0, Rty) packets, retrying until a DP(Prime) is successfully received. This case is not illustrated in the Figure above. The DPP shall contain a zero-length data payload.

DP(Stream x) - With this transition the host proposes the Stream ID *Stream x* to the device. If a DP with a Stream ID not equal to *Prime* is received from the host, the device shall transition to the **Move Data** state. The host may initiate this transition when it wishes to start a Stream transfer and is referred to as a *Host Initiated Move Data* or **HIMD**. A HIMD indicates the specific Stream that the Endpoint Data had been changed for. The device shall set *Stream x* to Ready due to this

transition. After entering the **Move Data** state, the device may reject the proposed Stream with an NRDY or accept the proposed Stream with an ACK TP. Upon transitioning to the **Move Data** state the device sets *CStream* to the value of the received Stream ID (*Stream x*). The DPP shall contain the first data payload for the Stream. Typically *Stream x* will be equal to the Stream ID (*Stream n*) in the last ERDY generated by the device. *Stream x* may not be equal to *Stream n* if one of the race conditions described below occurs, because the host drops ERDYs under these conditions.

DPH(Deferred) - If a DPH with the Deferred (DF) flag set is received, then the device shall transition to the **Deferred Prime Pipe** state. This packet may be received when the link has transitioned to a U1 or U2 state and the host has attempted a transition to **Prime Pipe** or **Move Data** (a HIMD).

8.12.1.4.3.5 Start Stream

In the **Start Stream** state, the device is waiting for the host to accept or reject the Active and Ready Stream selection that it has proposed.

DP(Stream n) - If a DP with a Stream ID equal to *Stream n* is received: the host has accepted the device's proposal for starting *Stream n* and provided the first packet of *Stream n* data, and the device shall transition to the **Move Data** state. Upon transitioning to the **Move Data** state the device sets *CStream* to the value of the received Stream ID (*Stream n*). The DPP shall contain the first data payload for *CStream*.

DP(NoStream, PP=0) - If a DP with a Stream ID equal to *NoStream* is successfully received, the host has rejected the device's proposal for starting *Stream n* and the device shall transition to the **Start Stream End** state. The DPP shall contain a zero-length data payload. The host shall reject a proposal from a device if there is no Endpoint Data available for the Stream. The device shall set *Stream n* to Not Ready due to this transition. Note, if an error is detected in the DP data the device shall remain in the **Start Stream** state, and issue ACK(NoStream, NumP>0, Rty) packets, retrying until a DP(NoStream) is successfully received. This case is not illustrated in the Figure above.

DP(Prime, PP=0) - If a DP with a Stream ID equal to *Prime* is received, a race condition has occurred. The host has entered the **Prime Pipe** state to inform the device that Endpoint Data for one or more Streams has been posted, at the same time that the device has attempted to initiate a Stream transfer, and their respective messages have passed each other on the link. The DPP shall contain a zero-length data payload. During this condition, the device is in the **Start Stream** state and the host is in the **Prime Pipe** state. To resolve this condition, the device shall transition to the **Prime Pipe** state. Note, if an error is detected in the DP data the device shall transition to the **Prime Pipe** state and perform any retries there.

DP(Stream x) - If a DP with a Stream ID not equal to *Stream n*, *Prime* or *NoStream* (e.g., equal to *Stream x*) is received, a race condition has occurred. The host has entered the **Move Data** state to initiate a transfer on *Stream x*, at the same time that the device has attempted to initiate a transfer on *Stream n*, and their respective messages have passed each other on the link. During this condition, the device is in the **Start Stream** state and the host is in the **Move Data** state. To resolve this condition, the device shall transition to the **Move Data** state. The device shall set *Stream x* to Ready due to this transition. Upon transitioning to the **Move Data** state the device sets *CStream* to the value of the received Stream ID (*Stream x*). The DPP shall contain the first data payload for *CStream*. The device may accept or reject the Stream proposed by the host when in the **Move Data** state.

DPH(Deferred) - If a DPH with the Deferred (DF) flag set is received, then the device shall transition the pipe to the **Idle** state. This packet is received when the link has transitioned to a U1 or U2 state while waiting for a host response to the Start Stream request. Note that this transition is highly unlikely because it can only occur if the tERDYTimeout has been exceeded. The device is expected to retry with an ERDY in this case. There is no DPP associated with a deferred DPH.

8.12.1.4.3.6 Start Stream End

In the **Start Stream End** state, the device has received a rejection of the Stream selection that it has proposed, and must respond to the DP from the host. The Bulk protocol requires an ACK or NRDY response for any DP sent. The Streams protocol specifies that an NRDY is sent.

NRDY(NoStream) - The device shall generate an NRDY with the Stream ID equal to *NoStream* and transition to the **Idle** state.

8.12.1.4.3.7 Move Data

In the Device OUT **Move Data** state, *CStream* is set to the same value at both ends of the pipe and the pipe may actively move data. The details of the bus transactions executed in the **Move Data** state and its exit conditions are defined in the Device OUT Move Data State Machine defined below.

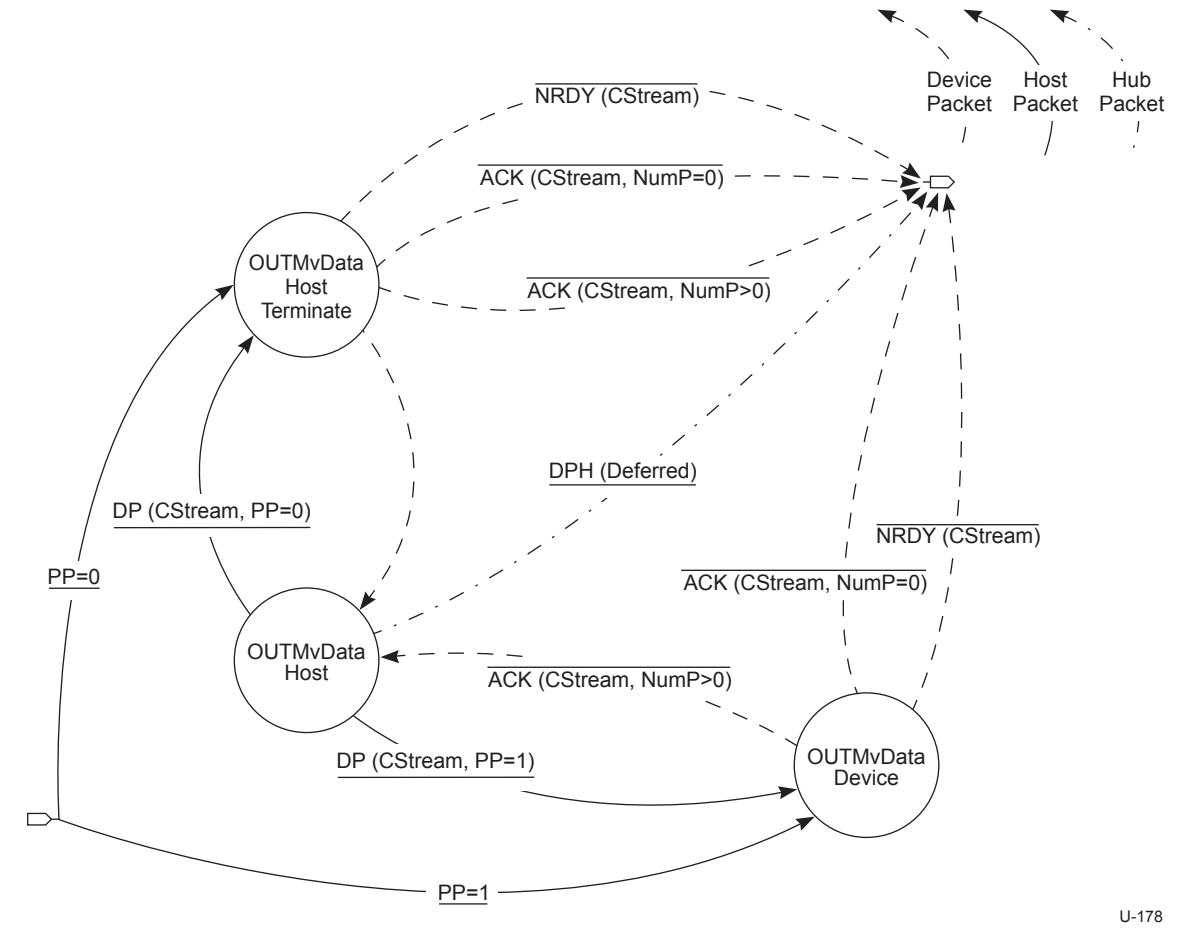


Figure 8-32. Device OUT Move Data State Machine (DOMDSM)

The Device OUT Move Data State Machine (DOMDSM) is entered from the **Start Stream** or **Idle** states as described above.

The DOMDSM allows either the device to terminate the Move Data operation because it has exhausted its Function Buffer space associated with a Stream or the host to terminate the Move Data operation because it has exhausted its Endpoint Data associated with a Stream.

PP=0 - Upon entry into the DOMDSM, if the host has only one packet of Endpoint Data available for the Stream then PP will equal 0 in the first DP received by the Device, and it shall transition to the **OUTMvData Device Terminate** state.

PP = 1 - Upon entry into the DOMDSM, if the host has more than one packet of Endpoint Data available for the Stream then PP will equal 1 in the first DP received by the Device, and it shall transition to the **OUTMvData Device** state.

The DOMDSM always exits to the **Idle** state. The Retry (Rty=1) flag shall never be set in a packet that causes a DOMDSM exit. A Stream pipe remains in the **Move Data** state during packet retries.

Note: The *Stream ID* value shall be *CStream* for all packets exchanged in the **Move Data** state. If a *Stream ID* value other than *CStream* is detected while in the DOMDSM the device should halt the endpoint.

Note: if *CStream* is not Active upon initially entering the **Move Data** state, the device may reject the Stream proposal with an NRDY or STALL the pipe, as defined by the associated Device Class.

8.12.1.4.3.8 OUTMvData Device

This state is initially entered from the **Start Stream** state or the **Idle** state. In this state the device acknowledges the last DP sent by the host or it may reject a HIMD from the host.

ACK(CStream, NumP>0) - If the device has more Function Buffer space available for *CStream*, then it shall send an ACK TP to the host with NumP > 0 and transition to the **OUTMvData Host** state. Note that the Retry (Rty) flag may be set in this packet if the device detected an error in the last DP from the host. The host shall continue the current burst until all retries are exhausted or a positive acknowledgement (Rty=0) is received. This transition shall indicate that the data payload of the previously received DP has been accepted by the endpoint for *CStream*.

ACK(CStream, NumP=0) - If the device has no more Endpoint Buffer space available for *CStream*, then it shall generate an ACK TP with NumP = 0, exit the DOMDSM and transition to the **Idle** state. This transition allows the device to exit from the **Move Data** state if its Endpoint Buffer space is exhausted. This transition shall indicate that the data payload of the previously received DP has been accepted by the endpoint for *CStream*.

NRDY(CStream) - The device may also terminate further *CStream* transfers by sending an NRDY with its Stream ID set to *CStream*, transitioning to the **Idle** state, exiting the DOMDSM. The device may generate this transition upon initial entry into the DOMDSM to reject a HIMD, or during a Stream transfer due to unexpected internal conditions where it wants to flow control *CStream*. This transition shall indicate that the data payload of the previously received DP has been dropped.

8.12.1.4.3.9 OUTMvData Host

In this state the host has just received an ACK TP from the device for a previous DP and has more Endpoint Data available for *CStream*. The host generates a DP in this state. The pipe will also wait in this state between bursts from the host.

DP(CStream, PP=1) - If the device receives a DP with PP = 1, then it shall transition to the **OUTMvData Device** state. The DPP shall contain a *CStream* data payload. This is the host response if it has more than one Max Packet Size of Endpoint Data available for *CStream*.

DP(CStream, PP=0) - If the device receives a DP with PP = 0, then it shall transition to the **OUTMvData Host Terminate** state. The DPP shall contain a *CStream* data payload. This is the host response if it has exhausted the Endpoint Data that it has available for *CStream*. The length of the DP will be less than or equal to one Max Packet Size.

DPH(Deferred) - If a DPH with the Deferred (DF) flag set is received, then the device shall transition to the **Idle** state, exiting the DOMDSM. This packet is received when the link has transitioned to a U1 or U2 state while waiting for the next DP from the host. There is no DPP associated with a deferred DPH.

8.12.1.4.3.10 OUTMvData Host Terminate

This state is entered because the host has just sent the last DP that it has available for *CStream*, e.g., it has exhausted its *CStream* Endpoint Data. In this state the device acknowledges the last DP from the host for the **Move Data** transfer.

ACK(CStream, NumP=0) - If the device has also exhausted its Function Buffer space, then it shall generate an ACK TP with NumP = 0 and transition to the **Idle** state, exiting the DOMDSM.

ACK(CStream, NumP>0) - If the device has not exhausted its Function Buffer space, then it shall generate an ACK TP with NumP > 0, and transition to the **Idle** state, exiting the DOMDSM. The device shall set *CStream* to Not Ready due to this transition.

ACK(CStream, NumP>0, Rty) - If an error was detected on the last DP by the device, then it shall generate an ACK TP with NumP > 0 and Rty = 1, so that the host will retry the last DP. The device shall then transition to the **OUTMvData Host** state.

NRDY(CStream) - The device may flow control on the last *CStream* transfer by sending an NRDY with its Stream ID set to *CStream* and transition to the **Idle** state, exiting the DOMDSM. The device may generate this transition due to unexpected internal conditions where it wants to flow control *CStream*.

8.12.1.4.4 Host IN Stream Protocol

This section defines the SuperSpeed packet exchanges that transition the host side of the Stream Protocol from one state to another on an IN bulk endpoint.

In the following text, a Host IN Stream state transition is assumed to occur at the point the host sends the first bit of the first symbol of a state machine related message to the device, or at the point the host first decodes state machine related message from the device.

For an IN pipe, Endpoint Buffers in the host receive Function Data from a device.

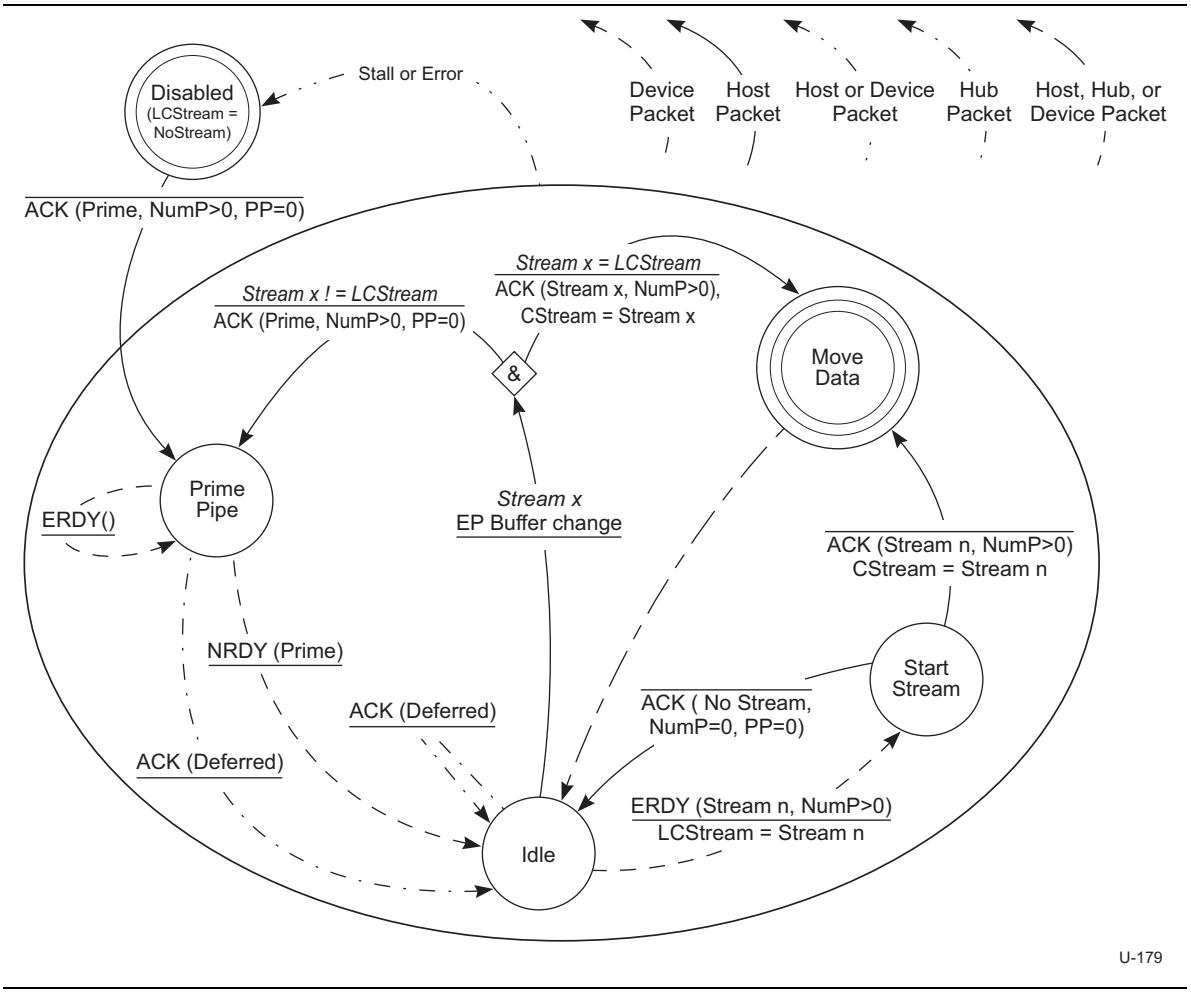


Figure 8-33. Host IN Stream Protocol State Machine (HISPSM)

8.12.1.4.4.1 Disabled

After an endpoint is configured or an endpoint error condition (Stall, tHostTransactionTimeout, etc.) request, the pipe is in the **Disabled** state and *LCStream* is initialized to *NoStream*.

ACK(Prime, NumP>0, PP=0) - When the initial Endpoint Buffers are assigned to the pipe by system software, the host shall send an ACK TP with the Stream ID field set to *Prime* to the device, and transition the pipe to the **Prime Pipe** state.

8.12.1.4.4.2 Prime Pipe

The **Prime Pipe** state informs the device that the Endpoint Buffers have been assigned to one or more Streams.

NRDY(Prime) – If the host receives an NRDY TP with its Stream ID field set to *Prime*, it shall transition to the **Idle** state. This transition is the normal termination of a Prime Pipe operation.

ACK(Deferred) - If an ACK with the Deferred (DF) flag set is received, then the host shall transition the pipe to the **Idle** state. This packet may be received when the link has transitioned to a

U1 or U2 state while the pipe was waiting for its initial Endpoint Buffer assignment. e.g. after an ACK(Prime, NumP>0, PP=0) has been generated in the **Disabled** state.

ERDY() - If an ERDY is received, a race condition has occurred. During this condition, the device is in the **Start Stream** state and the host is in the **Prime Pipe** state. The host has entered the **Prime Pipe** state to inform the device that the Endpoint Buffers for one or more Streams have been updated, at the same time that the device has attempted to initiate a Stream transfer, and their respective messages have passed each other on the link. To resolve this condition, the host shall remain in the **Prime Pipe** state and wait for an NRDY(Prime) from the device.

8.12.1.4.4.3 Idle

In the **Idle** state, the pipe is waiting for a Stream selection (e.g., a transition to **Start Stream** or **Move Data**) or a notification from the host that Stream Endpoint Data has been added or modified for the pipe (i.e., transition to **Prime Pipe**). Note that upon the initial entry in to **Idle** (i.e., from **Disabled**), only the device may initiate a Stream selection.

ERDY(Stream n, NumP>0) -If an ERDY is received, the host shall transition to the **Start Stream** state. The device generates an ERDY to select a specific Stream (*Stream n*) that it expects the host to begin IN transactions on. A device may initiate this transition when it wishes to start a Stream transfer, regardless of whether it had previously flow controlled the pipe or not. Note that the value of the ERDY NumP field reflects the amount of Endpoint Data the device has available for *Stream n*. The value of the ERDY NumP is informative and the method that a device uses for Stream selection is outside the scope of this specification and is normally defined by the Device Class associated with the pipe. Upon transitioning to the **Start Stream** state the host sets *LCStream* to the value of *Stream n*.

ACK(Deferred) - If an ACK with the Deferred (DF) flag set is received, then the host shall remain in the **Idle** state. This packet is received if the link has transitioned to a U1 or U2 state when the host rejects a Start Stream request from the device (i.e., due to an ACK(NoStream, NumP=0, PP=0)). This case only occurs if tERDYTimeout is exceeded.

Stream x EP Buffer Change - This transition occurs if the state of one or more Endpoint Buffers has changed in the host. The host evaluates (at the Joint "**&**") the ID of the Stream that software presents to the host controller (*Stream x*) and transitions to the **Prime Pipe** or **Move Data** states. This is an optimization that allows the host to transition the Stream pipe directly to the **Move Data** state, rather than going through the **Prime Pipe**, **Start Stream**, **Move Data** sequence, and is referred to as a *Host Initiated Move Data* or **HIMD**. The specific algorithm used to make this decision is host specific.

&

ACK(Prime, NumP>0, PP=0) - If the transition to **Prime Pipe** is selected, then the host shall generate a ACK TP with the Stream ID = *Prime*, NumP > 0, and PP = 0, and transition to the **Prime Pipe** state. Note that the host asserts a non-zero NumP value so that the device may respond with an NRDY. If NumP = 0, the device would consider it a Terminating ACK and not respond. Typically the **Prime Pipe** transition will be selected when the Stream that has just had its host Endpoint Buffers modified is not the same Stream that the device has last selected, e.g., *Stream x != LCStream*.

ACK(Stream x, NumP>0) - If the transition to **Move Data** is selected, then the host shall generate a ACK TP with the Stream ID = *Stream x* and NumP > 0, and transition to the **Move**

Data state. Upon transitioning to the **Move Data** state the host sets *CStream* to the value of *Stream x*. Typically the **Move Data** transition will be selected when the Stream that has just had its Endpoint buffers modified is the same Stream as the one that the device last selected, e.g., *Stream x = LCStream*. PP shall equal 1 because the host is capable of receiving another DP from the device. This transition optionally may be disabled in some hosts, and some Device Classes may not process this transition (e.g., Mass Storage UASP).

8.12.1.4.4.4 Start Stream

In the **Start Stream** state, the device has sent an ERDY proposing to the host that it initiate an IN transfer for *Stream n* and it is waiting for the host to accept or reject the Stream selection.

ACK(Stream n, NumP>0) - If the host has accepted the device's proposal for starting *Stream n*, then it shall transmit an ACK TP with a Stream ID equal to *Stream n*, and transition to the **Move Data** state. Upon transitioning to the **Move Data** state the host sets *CStream* to the value of *Stream n*. The host shall accept a Stream proposal from a device if there are Endpoint Buffers available to receive the Function Data for the Stream. PP shall equal 1 because the host is capable of receiving another DP from the device.

ACK(NoStream, NumP=0, PP = 0) - If the host rejects the device's proposal for starting *Stream n*, then it shall transmit an ACK TP with a Stream ID equal to *NoStream*, and transition to the **Idle** state. The host shall reject a Stream proposal from a device if there are no Endpoint Buffers available to receive the Function Data for the Stream.

8.12.1.4.4.5 Move Data

In the Host IN **Move Data** state, *CStream* is set to the same value at both ends of the pipe and the pipe may actively move data. The details of the bus transactions executed in the **Move Data** state and its exit conditions are defined in the Host IN Move Data State Machine defined below.

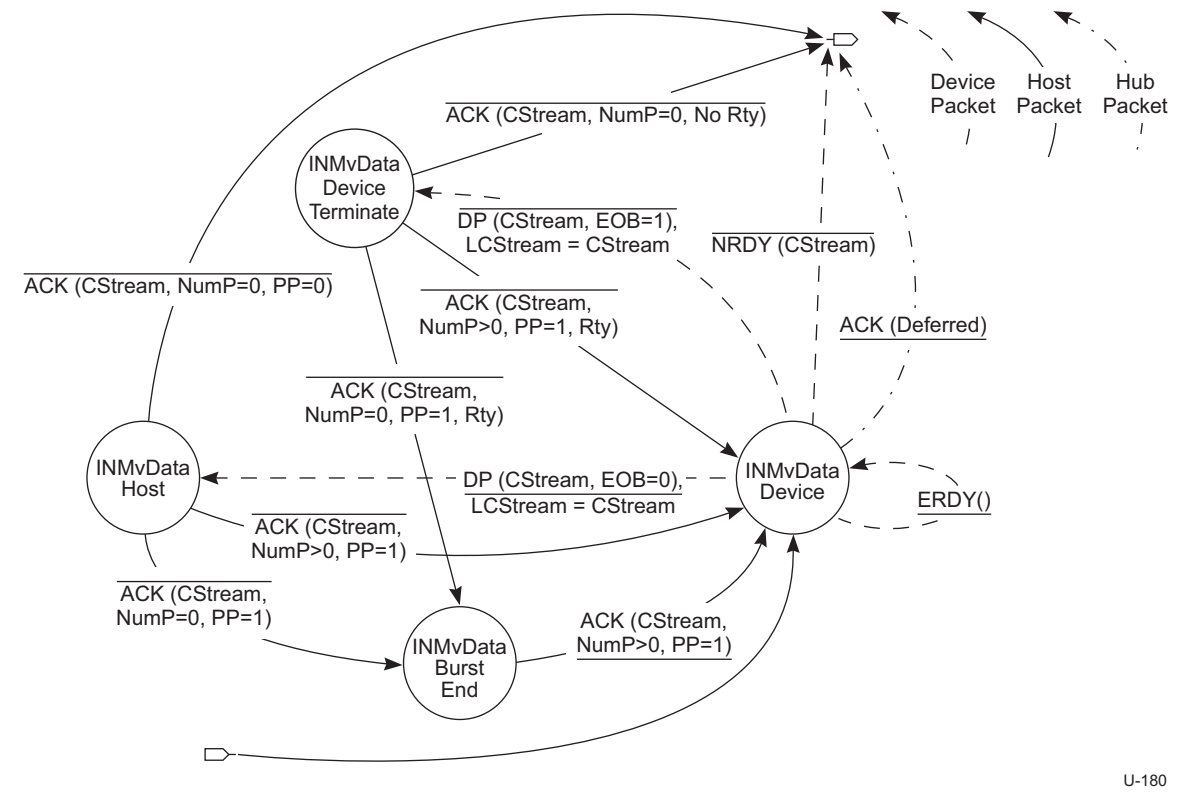


Figure 8-34. Host IN Move Data State Machine (HIMDSM)

The Host IN Move Data State Machine (HIMDSM) is entered from the **Start Stream** or **Idle** states as described above. The entry into the HIMDSM immediately transitions to the **INMvData Device** state. The HIMDSM allows either the device to terminate the Move Data operation because it has exhausted its Function Data associated with a Stream or the host to terminate the Move Data operation because it has exhausted its Endpoint Buffer space associated with a Stream.

The HIMDSM always exits to the **Idle** state. The Retry (Rty=1) flag shall never be set in a packet that causes a HIMDSM exit. A Stream pipe remains in the Move Data state during packet retries.

Note: The *Stream ID* value shall be *CStream* for all packets exchanged in the **Move Data** state, except the **INMvData Device** substate ERDY transition. For the identified substates, if a *Stream ID* value other than *CStream* is detected while in the HIMDSM the host should halt the endpoint.

8.12.1.4.4.6 INMvData Device

This state is initially entered from the **Start Stream** state or the **Idle** state. In this state the host is waiting for a DP from the device or a rejection of a HIMD.

DP(CStream, EOB=0) - If the host receives a DP with EOB = 0, it shall copy the DP data to the Endpoint Buffer associated with the Stream and transition to the **INMvData Host** state. The DPP shall contain a *CStream* data payload. This transition occurs when the device returns IN data and has more Function Data to send. Upon transitioning to the **INMvData Host** state the host sets *LCStream* to the value of *CStream*. This action updates *LCStream* with the value of *CStream* if the device accepts a HIMD, i.e., *LCStream* records the last Stream that was of interest to the device.

DP(CStream, EOB=1) - If the host receives a DP with EOB = 1, it shall copy the DP data to the Endpoint Buffer associated with the Stream and transition to the **INMvData Device Terminate** state. The DPP shall contain a *CStream* data payload. This transition occurs when device returns IN data and has no more Function Data to send, e.g., it is terminating the Move Data operation because this DP exhausts the Function Data available for this Stream. Upon transitioning to the **INMvData Device Terminate** state the host sets *LCStream* to the value of *CStream*. This action updates *LCStream* with the value of *CStream* if the device accepts a HIMD, i.e., *LCStream* records the last Stream that was of interest to the device.

NRDY(CStream) - If the host receives an NRDY, it shall exit the HIMDSM and transition to the **Idle** state. This transition may occur upon initial entry into the HIMDSM when the device rejects a HIMD, or during a Stream transfer due to unexpected internal device conditions where it wants to flow control *CStream*.

ACK(Deferred) - If the host receives an ACK with the Deferred (DF) flag set, then it shall exit the HIMDSM and transition to the **Idle** state. This packet shall be received if a link in the path between the host and the device has transitioned to a U1 or U2 state. There are two cases when this transition may occur: 1) the host has attempted a HIMD, and 2) between bursts. Case 1 is likely to occur if there has been a long host delay in obtaining buffers for the Stream. Case 2 may occur if there is a lot of endpoint activity on other devices delaying the time between bursts. The device treats this transition like a Prime Pipe and will send an ERDY to restart the stream when it receives the Deferred ACK forwarded to it by a hub.

ERDY() - If an ERDY is received, a race condition has occurred. During this condition, the device is in the **Start Stream** state and the host is in the **Move Data** state. The host has entered the **Move Data** state as the result of a HIMD, at the same time that the device has attempted to initiate a Stream transfer, and their respective messages have passed each other on the link. To resolve this condition, the host shall remain in the **INMvData Device** state and wait for a DP or an NRDY from the device.

8.12.1.4.4.7 INMvData Host

In this state the host has received a DP from a device that has more Function Data available for *CStream*. The host responds with an acknowledgement after copying the received data to the Endpoint Buffer space associated with the Stream.

ACK(CStream, NumP>0, PP=1) - If more Endpoint Buffer space is available for the Stream and the host is continuing the current burst to the device, then the host shall generate an ACK TP with NumP > 0 and PP = 1, and transition to the **INMvData Device** state. If the host detected an error on the last DP from the device, then the Rty flag shall be set. The host may continue the **INMvData Host** to **INMvData Device** loop until all retries are exhausted or a good packet is received by the device. If the current burst terminates before all retries are exhausted, the host may transition to the **INMvData Burst End** state (with Rty=1) and return to the INMvData Device state (with Rty=1) at the next available opportunity to continue the retry process within the constraints of the endpoint type.

ACK(CStream, NumP=0, PP=1) - If more Endpoint Buffer space is available for the Stream, however the host must terminate the current burst to the device, then the host shall generate an

ACK TP with NumP = 0 and PP = 1, and transition to the **INMvData Burst End** state. If the host detected an error on the last DP from the device, then the Rty flag shall be set.

ACK(CStream, NumP=0, PP=0) - If the host did not detect an error on the last DP received from the device and the Endpoint Buffer space available for the Stream is exhausted, then the host shall generate an ACK TP with NumP = 0 and PP = 0, and transition to the **Idle** state, exiting the HIMDSM. This transition informs the device the host has exhausted its Endpoint Buffer space for the Stream.

8.12.1.4.4.8 INMvData Burst End

This state is entered because the host has terminated a burst on a stream pipe. The host will exit this state when it is ready to start another burst. If this state was entered while retrying (Rty =1), then the host shall continue the retry process within the constraints of the endpoint when exiting the state.

ACK(CStream, NumP>0, PP=1) - When ready to start another burst to the device on *CStream*, the host shall generate an ACK with NumP > 0 and PP = 1 and transition to the **INMvData Device** state. Note, if the Rty flag was set when the state was entered, then it shall be set upon exit.

8.12.1.4.4.9 INMvData Device Terminate

In this state the host has received the last DP from a device for this Move Data operation because the device has exhausted the Function Data it has available for *CStream*. The host responds with an acknowledgement after copying the received data to the Endpoint Buffer space associated with the Stream and exits the HIMDSM. If the DP received from the device is bad, then retries may be performed within the constraints of the endpoint type.

ACK(CStream, NumP=0, No Rty) - If the DP received from the device is good, then the host generates an ACK with NumP = 0 and Rty = 0, and transitions to the **Idle** state, exiting the HIMDSM.

ACK(CStream, NumP>0, PP=1, Rty) - If the DP received from the device is bad and the current burst is not complete, then the host shall generate an ACK with NumP > 0, PP = 1 and Rty = 1, and transition to the **INMvData Device** state. The host may continue the **INMvData Device Terminate** to **INMvData Device** loop until all retries are exhausted or a good packet is received.

ACK(CStream, NumP=0, PP=1, Rty) - If the DP received from the device is bad and the current burst is complete, then the host shall generate an ACK with NumP = 0, PP = 1 and Rty = 1, and transition to the **INMvData Burst End** state. The host shall continue the retry process in the next burst.

8.12.1.4.5 Host OUT Stream Protocol

This section defines the SuperSpeed packet exchanges that transition the host side of the Stream Protocol from one state to another on an OUT bulk endpoint.

In the following text, a Host OUT Stream state transition is assumed to occur at the point the host sends the first bit of the first symbol of a state machine related message to the device, or at the point the host first decodes state machine related message from the device.

For an OUT pipe, Function Buffers in the device receive Endpoint Data from the host.

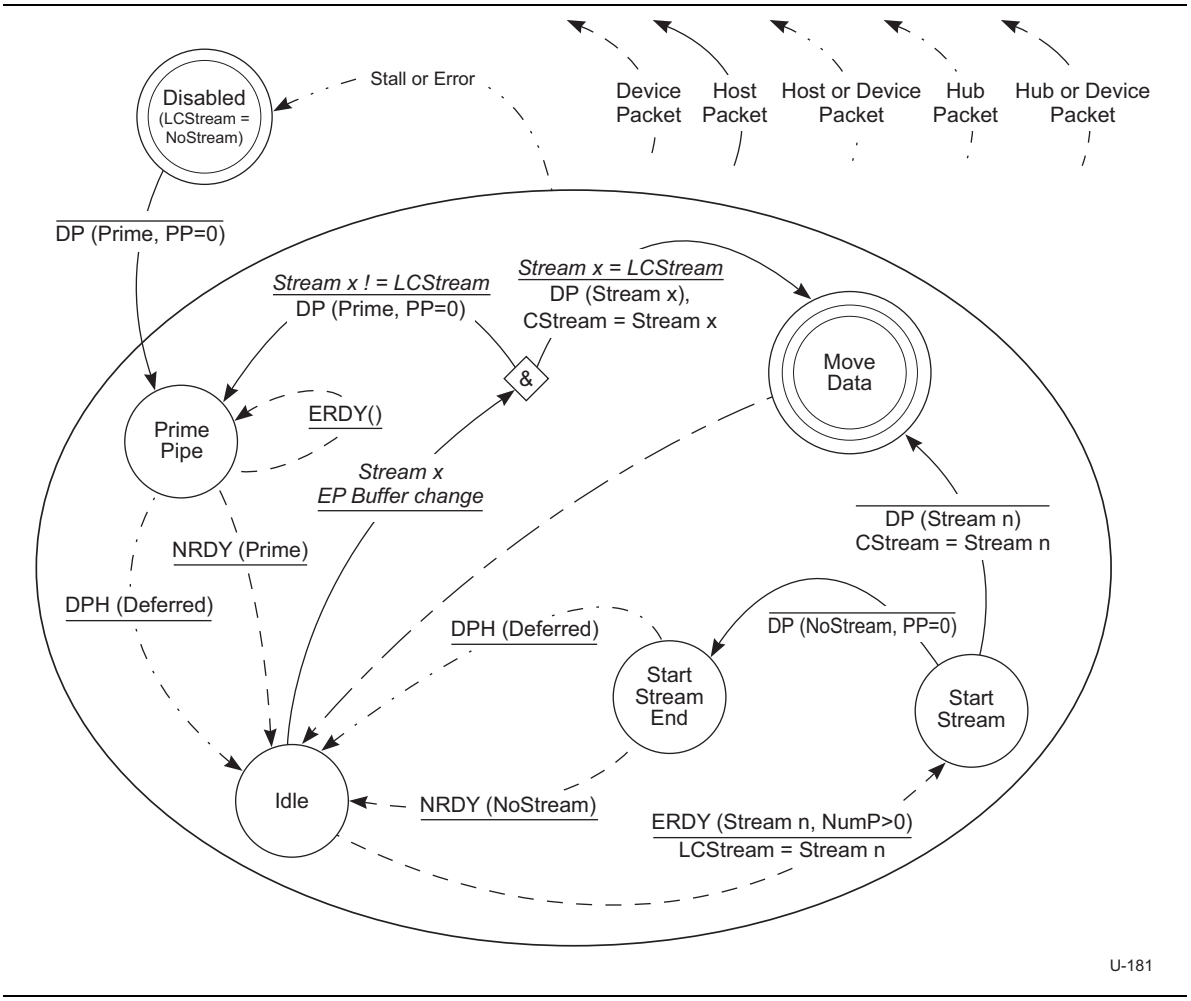


Figure 8-35. Host OUT Stream Protocol State Machine (HOSPSM)

8.12.1.4.5.1 Disabled

After an endpoint is configured or receives a SetFeature(ENDPOINT_HALT) request, the pipe is in the **Disabled** state and *LCStream* is initialized to *NoStream*.

DP(Prime, PP=0) - When the initial Endpoint Data is assigned to the pipe by system software, the host shall send a zero-length DP with the Stream ID field set to *Prime* to the device, and transition the pipe to the **Prime Pipe** state. The DPP shall contain a zero-length data payload.

8.12.1.4.5.2 Prime Pipe

The **Prime Pipe** state informs the device that Endpoint Buffers have been assigned to one or more Streams. Note, this state is entered when the host transmits a DP(Prime) from the **Disabled** or the **Idle** state. If an error is detected in the DP data by the device, the device shall issue ACK(Prime, NumP>0, Rty) packet, retrying until a DP(Prime) is successfully received. The host may retransmit the DP(Prime) and shall remain in the **Prime Pipe** state until the device successfully receives the DP(Prime) and returns an NRDY(Prime), or the retries for the pipe are exhausted and the host halts the pipe. This case is not illustrated in the Figure above.

NRDY(Prime) - If the host receives an NRDY TP with its Stream ID field set to *Prime*, it shall transition to the **Idle** state. This transition is the normal termination of a Prime Pipe operation.

DPH(Deferred) - If a DPH with the Deferred (DF) flag set is received, then the host shall transition the pipe to the **Idle** state. This packet may be received when the link has transitioned to the U1 or U2 state while the pipe waiting for its initial Endpoint Buffer assignment, e.g., after a DP(Prime, PP=0) has been generated in the **Disabled** state. There is no DPP associated with a deferred DPH.

ERDY() - If an ERDY is received, a race condition has occurred. During this condition, the device is in the **Start Stream** state and the host is in the **Prime Pipe** state. The host has entered the **Prime Pipe** state to inform the device that the Endpoint Buffers for one or more Streams have been updated, at the same time that the device has attempted to initiate a Stream transfer, and their respective messages have passed each other on the link. To resolve this condition, the host shall remain in the **Prime Pipe** state and wait for an NRDY from the device.

8.12.1.4.5.3 Idle

In the **Idle** state, the pipe is waiting for a Stream selection (e.g., a transition to **Start Stream** or **Move Data**) or a notification from the host that Stream Endpoint Data has been added or modified for the pipe (i.e., transition to **Prime Pipe**). Note that upon the initial entry in to **Idle** (i.e., from **Disabled**), only the device may initiate a Stream selection.

ERDY(Stream n, NumP>0) - If an ERDY is received, the host shall transition to the **Start Stream** state. The device generates an ERDY to select a specific Stream (*Stream n*) that it expects the host to begin OUT transactions on. A device may initiate this transition when it wishes to start a Stream transfer, regardless of whether it had previously flow controlled the pipe or not. Note that the value of the ERDY NumP field reflects the amount of Endpoint Buffer space the device has available for *Stream n*. The method that a device uses for Stream selection is outside the scope of this specification and is normally defined by the Device Class associated with the pipe. Upon transitioning to the **Start Stream** state the host sets *LCStream* to the value of *Stream n*.

Stream x EP Buffer Change - This transition occurs if Endpoint Data has been posted for one or more Streams in the host. The host evaluates (at the Joint "**&**") the ID of the Stream that software presents to the host controller (*Stream x*) and transitions to the **Prime Pipe** or **Move Data** states. This is an optimization that allows the host to transition the Stream pipe directly to the **Move Data** state, rather than going through the **Prime Pipe**, **Start Stream**, **Move Data** sequence, and is referred to as a *Host Initiated Move Data* or **HIMD**. The specific algorithm used to make this decision is host specific.

&

DP(Prime, PP=0) - If the transition to **Prime Pipe** is selected, then the host shall generate a DP with the Stream ID = *Prime* and PP = 0, and transition to the **Prime Pipe** state. The DPP shall contain a zero-length data payload. Typically the **Prime Pipe** transition will be selected when the Stream that has just had its Endpoint Data modified is not the same Stream that the device has last selected, e.g., *Stream x != LCStream*.

DP(Stream x) - If the transition to **Move Data** is selected, then the host shall generate a DP with the Stream ID = *Stream x*, and transition to the **Move Data** state. Upon transitioning to the **Move Data** state the host sets *CStream* to the value of *Stream x*. The DPP shall contain the first data payload for *CStream*. Typically the **Move Data** transition will be selected when the Stream that has just had its Endpoint Data modified is the same Stream as the one that the

device last selected, e.g., *Stream x = LCStream*. The value of PP shall depend on the amount of Endpoint data the host has available. If the host has more than Max Packet Size Endpoint Data available for the Stream, then PP = 1 else PP = 0. This transition may be optionally be disabled in some hosts, and some Device Classes may not process this transition (e.g., Mass Storage UASP).

8.12.1.4.5.4 Start Stream

In the **Start Stream** state, the device has sent an ERDY proposing to the host that it initiate an OUT transfer for *Stream n* and it is waiting for the host to accept or reject the Stream selection.

DP(Stream n) - If the host has accepted the device's proposal for starting *Stream n*, then it shall transmit a DP with a Stream ID equal to *Stream n*, and transition to the **Move Data** state. Upon transitioning to the **Move Data** state the host sets *CStream* to the value of *Stream n*. The DPP shall contain the first data payload for *CStream*. The host shall accept a Stream proposal from a device if there is Endpoint Data available for the Stream.

DP(NoStream, PP = 0) - If the host rejects the device's proposal for starting *Stream n*, then it shall transmit a DP with a Stream ID equal to *NoStream*, and transition to the **Start Stream End** state. The DPP shall contain a zero-length data payload. The host shall reject a Stream proposal from a device if there is no Endpoint Data available to send for the Stream.

8.12.1.4.5.5 Start Stream End

In the **Start Stream End** state, the host has rejected a proposed Stream ID from the device because there was no Endpoint Data available for *Stream n*. Note, this state is entered when the host transmits a DP(NoStream) from the **Start Stream** state. If an error is detected in the DP data by the device, the device shall issue ACK(NoStream, NumP>0, Rty) packet, retrying until a DP(NoStream) is successfully received. The host may retransmit the DP(NoStream) and shall remain in the **Start Stream End** state until the device successfully receives the DP(NoStream) and returns an NRDY(NoStream), or the retries for the pipe are exhausted and the host halts the pipe. This case is not illustrated in the Figure above.

NRDY(NoStream) - If an NRDY with the Stream ID equal to *NoStream* is received, the host shall transition to the **Idle** state.

DPH(Deferred) - If a DPH with the Deferred (DF) bit set is received, the host shall transition to the **Idle** state. This packet is received when the link has transitioned to a U1 or U2 state while waiting for a host response to the Start Stream request. Note that this transition can occur only if the tERDYTimeout has been exceeded. There is no DPP associated with a deferred DPH.

8.12.1.4.5.6 Move Data

In the Host OUT **Move Data** state, *CStream* is set to the same value at both ends of the pipe and the pipe may actively move data. The details of the bus transactions executed in the **Move Data** state and its exit conditions are defined in the Host OUT Move Data State Machine defined below.

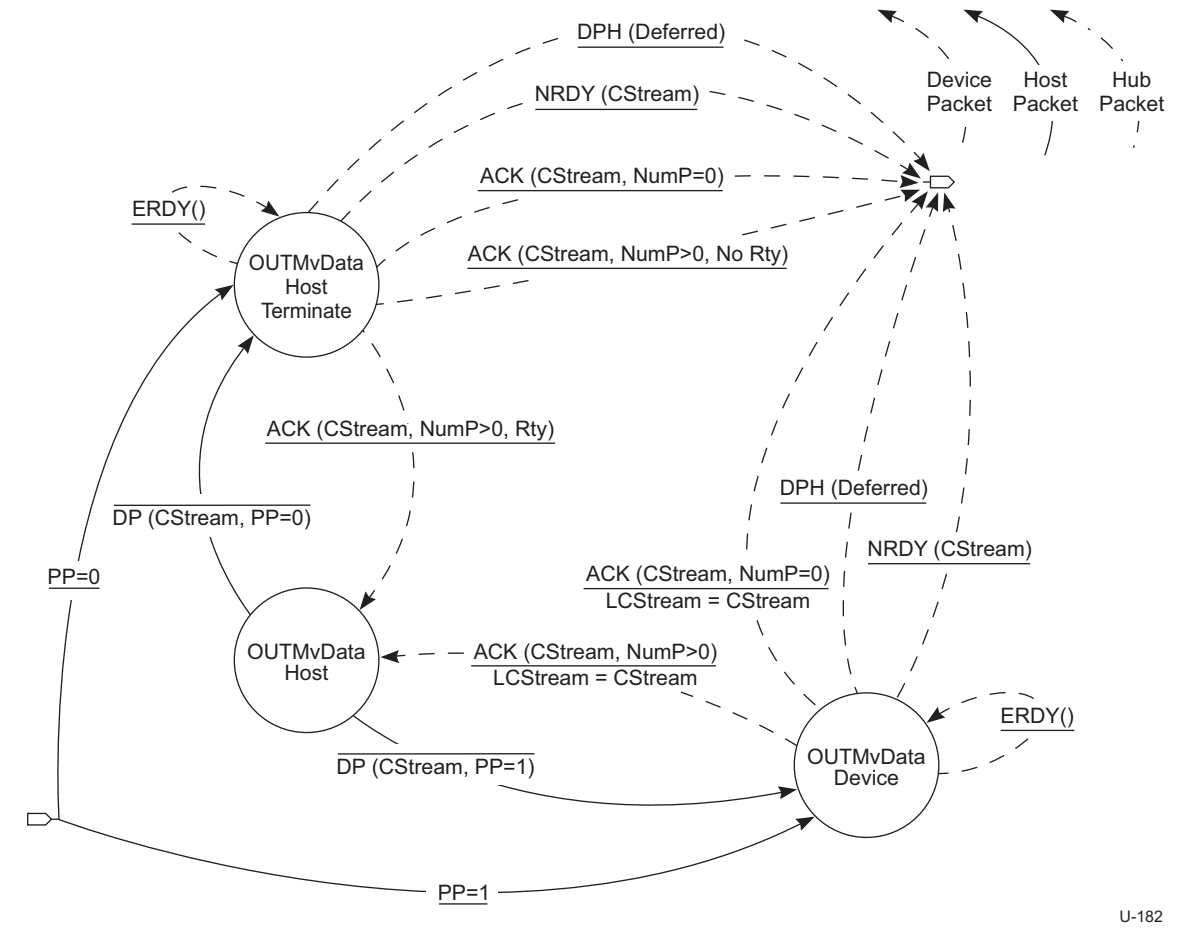


Figure 8-36. Host OUT Move Data State Machine (HOMDSM)

The Host OUT Move Data State Machine (HOMDSM) is entered from the **Start Stream** or **Idle** states as described above. The entry into the HOMDSM immediately transitions to the **OUTMvData Device** state. The HOMDSM allows either the device to terminate the Move Data operation because it has exhausted its Function Buffer space associated with a Stream or the host to terminate the Move Data operation because it has exhausted its Endpoint Data associated with a Stream.

PP = 0 - Upon entry into the HOMDSM, if the host has only one packet of Endpoint Data available for the Stream then PP will equal 0 in the first DP sent to the Device, and it shall transition to the **OUTMvData Device Terminate** state.

PP = 1 - Upon entry into the HOMDSM, if the host has more than one packet of Endpoint Data available for the Stream then PP will equal 1 in the first DP sent to the Device, and it shall transition to the **OUTMvData Device** state.

The HOMDSM always exits to the **Idle** state. The Retry (Rty=1) flag shall never be set in a packet that causes a HOMDSM exit. A Stream pipe remains in the Move Data state during packet retries.

Note: The *Stream ID* value shall be *CStream* for all packets exchanged in the **Move Data** state, except the **OUTMvData Device** substate ERDY transition. For the identified substates, if a *Stream ID* value other than *CStream* is detected while in the HOMDSM the host should halt the endpoint.

8.12.1.4.5.7 OUTMvData Device

This state is initially entered from the **Start Stream** state or the **Idle** state. In this state the host is waiting for an ACK TP or a rejection of a HIMD from the device.

ACK(CStream, NumP>0) - If the host receives an ACK TP with NumP > 0, it shall transition to the **OUTMvData Host** state. This transition occurs when device has more Function Buffer space available for the stream. If the device detected an error on the last DP from the host, then the Retry (Rty) flag shall be set. If a Retry is requested, the host shall continue the current burst until all retries are exhausted or a good packet is transmitted. The host shall set *LCStream = CStream*. This action updates *LCStream* with the value of *Stream x* if the device accepts a HIMD, i.e., *LCStream* records the last Stream that was of interest to the device.

ACK(CStream, NumP=0) - If the host receives an ACK TP with NumP = 0, it shall transition to the **Idle** state, exiting the HOMDSM. This transition occurs when device has no more Function Buffer space available for the Stream, e.g., it is terminating the Move Data operation because the last DP exhausted its Function Buffer space. The host shall set *LCStream = CStream*. This action updates *LCStream* with the value of *Stream x* if the device accepts a HIMD, i.e., *LCStream* records the last Stream that was of interest to the device.

NRDY(CStream) - If the host receives an NRDY, it shall transition to the **Idle** state, exiting the HOMDSM. This transition may occur upon initial entry into the HOMDSM when the device rejects a HIMD, or during a Stream transfer due to unexpected internal device conditions where it wants to flow control *CStream*.

DPH(Deferred) - If the host receives a DPH with the Deferred (DF) flag set, then it shall transition to the **Idle** state, exiting the HOMDSM. This packet may be received when the link has transitioned to a U1 or U2 state and the host has attempted a HIMD or between bursts on the OUT pipe, if there is a lot of endpoint activity on other devices and the Ux Timeouts in the path to this device are set to short values. When this transition occurs the host will wait in the **Idle** state for an ERDY from the device to restart the stream. There is no DPP associated with a deferred DPH.

ERDY() - If an ERDY is received, a race condition has occurred. During this condition, the device is in the **Start Stream** state and the host is in the **Move Data** state. The host has entered the **Move Data** state as the result of a HIMD, at the same time that the device has attempted to initiate a Stream transfer, and their respective messages have passed each other on the link. To resolve this condition, the host shall remain in the **OUTMvData Device** state and wait for an ACK or an NRDY from the device.

8.12.1.4.5.8 OUTMvData Host

In this state the host has received an ACK TP from a device and the device has more Function Buffer space available for *CStream*. The host responds with a DP containing Endpoint Data associated with the Stream. The pipe will also wait in this state between bursts from the host. Note, that the DP retry process may span bursts.

DP(CStream, PP=1) - If more Endpoint Data is available for the Stream and the host is continuing the current burst to the device, then the host shall generate a DP with PP = 1, and transition to the

OUTMvData Device state. The DPP shall contain a *CStream* data payload. If the Rty flag was set in the last ACK from the device, then the host shall resend the appropriate DP until all retries are exhausted or a good DP is acknowledged by the device.

DP(CStream, PP=0) - If the Endpoint Data available for the Stream is exhausted by transmitting this DP, then the host shall generate a DP with PP = 0, and transition to the **OUTMvData Host Terminate** state. The DPP shall contain a *CStream* data payload. This transition informs the device the host has exhausted its Endpoint Data for the Stream.

8.12.1.4.5.9 OUTMvData Host Terminate

In this state the host has just exhausted the Endpoint Data that it has available for *CStream* and sent the last DP for the Stream. The host is waiting for an acknowledgement for the last DP of the Stream.

ACK(CStream, NumP=0) - If the host receives and ACK TP with NumP = 0 and Rty = 0, then the host shall transition to the **Idle** state, exiting the HOMDSM. This transition occurs when the device has successfully received the last DP, and both the host and the device have exhausted their respective Endpoint Data and Function Buffer space at the same time.

ACK(CStream, NumP>0, No Rty) - If the host receives an ACK TP with NumP > 0, PP = 0, and Rty = 0, then the host shall transition to the **Idle** state, exiting the HOMDSM. This transition occurs when the device has successfully received the last DP, and the host has exhausted its Endpoint Data for the *CStream*, but the device still has more Function Buffer space available.

ACK(CStream, NumP>0, Rty) - If the host receives an ACK TP with NumP > 0 and Rty = 1, then the host shall transition the **OUTMvData Host** state and resend the appropriate DP. This transition occurs when the last packet received by the device was bad, and a Retry is required. The host shall continue the **OUTMvData Host Terminate** to **OUTMvData Host** loop until all retries are exhausted or a good DP is acknowledged by the device.

NRDY(CStream) - If the host receives an NRDY, it shall transition to the **Idle** state, exiting the HOMDSM. This transition may occur during a Stream transfer due to unexpected internal device conditions where it wants to flow control *CStream*.

DPH(Deferred) - If a DPH with the Deferred (DF) flag set is received, then the host shall transition to the **Idle** state, exiting the HOMDSM. This packet is received when the link had transitioned to the U1 or U2 state before the last DP was sent by the host. There is no DPP associated with a deferred DPH.

8.12.2 Control Transfers

Control transfers have a minimum of two transaction stages: Setup and Status. A control transfer may optionally contain a Data stage between the Setup and Status stages. The direction of the Data stage is indicated by the **bmRequestType** field which is present in the first byte of the data payload of the Setup packet. During the Setup stage, a SETUP transaction is used to transmit information to a control endpoint of the device. SETUP transactions are similar in format to a Bulk OUT transaction but have the **Setup** field set to one in the DPH along with the **Data Length** field set to eight. In addition, the Setup packet always uses a Data sequence number of zero. A device receiving a Setup packet shall respond as defined in Section 8.11.4. The **Direction** field shall be set to zero in TPs or DPs exchanged between the host and any control endpoint on the device regardless of the stage or direction of the control transfer.

Note that if the endpoint successfully received the SETUP packet, it may return an ACK TP with the **NumP** field set to zero if it wants to flow control the control transfer. A device shall send an ERDY when it is ready to resume the control transfer (either the Data or Status stage). Note that an endpoint may return an ACK TP with the **NumP** field set to zero in response to a SETUP packet if it wants to flow control the control transfer. A device must send an ERDY to start the Data or Status stage. Note that the host may resume transactions to any endpoint – even if the endpoint had not returned an ERDY TP after returning a flow control response.

The Data stage, if present, of a control transfer consists of one or more IN or OUT transactions and follows the same protocol rules as bulk transfers except that the **Direction** field shall always be set to zero. The Data stage always starts with the sequence number set to zero. All the transactions in the Data stage shall be in the same direction (i.e., all INs or all OUTs). The maximum amount of data to be sent during the data stage and its direction are specified during the Setup stage. If the amount of data exceeds the data packet size, the data is sent in multiple data packets that carry the maximum packet size. Any remaining data is sent as a residual in the last data packet.

Note that all control endpoints only support a burst of one and hence the host can only send or receive one packet at a time to or from a control endpoint.

The Status stage of a control transfer is the last transaction in the sequence. The status stage transaction is identified by a TP with the SubType set to *STATUS*. In response to a STATUS TP with zero in the **Deferred** bit, a device shall send an NRDY, STALL, or ACK TP. If a device sends an NRDY TP, the host shall wait for it to send an ERDY TP for that control endpoint before sending another STATUS TP to the device. However the host may resume transactions to any endpoint – even if the endpoint had not returned an ERDY TP after returning a flow control response. If the **Deferred** bit is set in the STATUS TP, then the device shall send an ERDY TP to indicate to the host that is ready to complete the status stage of the control transfer.

Figure 8-37 and Figure 8-38 show the transaction order, the data sequence number value, and the data packet types for control read and write sequences.

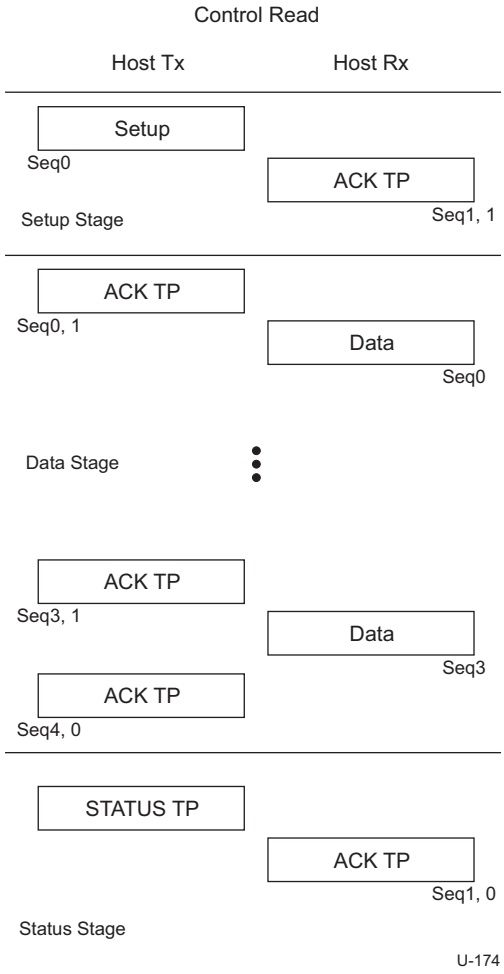


Figure 8-37. Control Read Sequence

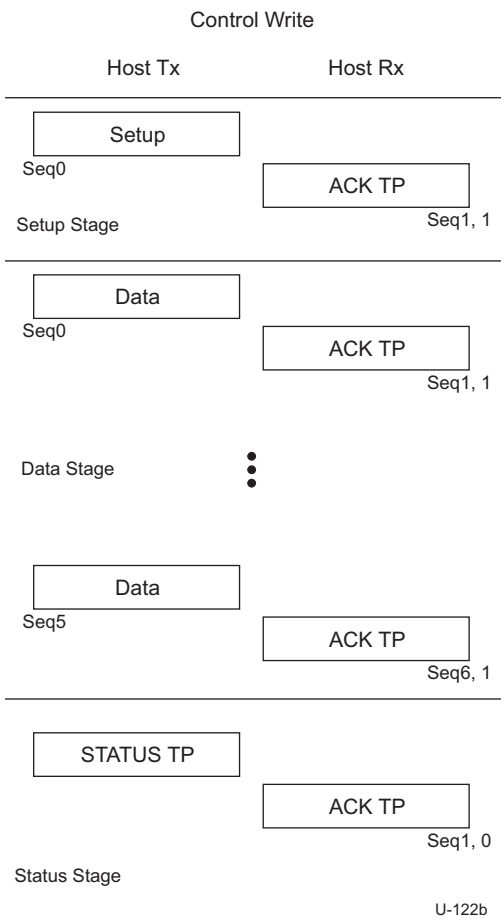


Figure 8-38. Control Write Sequence

When a STALL TP is sent by a control endpoint in either the Data or Status stages of a control transfer, a STALL TP shall be returned on all succeeding accesses to that endpoint until a SETUP DP is received. An endpoint shall return an ACK TP when it receives a subsequent SETUP DP. For control endpoints, if an ACK TP is returned for the SETUP transaction, the host expects that the endpoint has automatically recovered from the condition that caused the STALL and the endpoint shall operate normally.

8.12.2.1 Reporting Status Results

During the Status stage, a device reports to the host the outcome of the previous Setup and Data stages of the transfer. Three possible results may be returned:

- The command sequence completed successfully.
- The command sequence failed to complete.
- The device is still busy completing the command.

Status reporting is always in the device-to-host direction. Table 8-29 summarizes the type of responses required for each. All Control transfers return status in the TP that is returned to the host in response to a STATUS TP transaction.

Note that even though the status reporting is always in the device-to-host direction, the STATUS TP shall be treated as an OUT transaction. A host may start performing IN transactions to another endpoint without waiting for the response for the STATUS TP.

Table 8-29. Status Stage Responses

Status Response	TP Sent by Device
Request completes	ACK TP
Request has an error	STALL TP
Device is busy	NRDY TP

The host shall send a STATUS TP to the control pipe to initiate the Status stage. The pipe's handshake response to this TP indicates the current status. An NRDY TP indicates that a device is still processing the command and that the device shall send an ERDY TP when it completes the command. An ACK TP indicates that a device has completed the command and is ready to accept a new command. A STALL TP indicates that a device has an error that prevents it from completing the command.

The **NumP** field of the ACK TP sent by a control endpoint on the device shall be set to zero. However this is not considered a flow control condition for a control endpoint.

If during a Data stage a control pipe is sent more data or is requested to return more data than was indicated in the Setup stage, it shall return a STALL TP. If a control pipe returns a STALL TP during the Data stage, there shall not be a Status stage for that control transfer.

8.12.2.2 Variable-length Data Stage

A control pipe may have a variable-length data phase in which the host requests more data than is contained in the specified data structure. When all of the data structure is returned to the host, a device indicates that the Data stage is ended by returning a DP that has a payload less than the maximum packet size for that endpoint.

Note that if the amount of data in the data structure that is returned to the host is less than the amount requested by the host and is an exact multiple of maximum packet size then a control endpoint shall send a zero length DP to terminate the data stage.

8.12.2.3 STALL TPs Returned by Control Pipes

Control pipes have the unique ability to return a STALL TP due to problems in control transfers. If a device is unable to complete a command, it returns a STALL TP in the Data and/or Status stages of the control transfer. Unlike the case of a functional stall, protocol stall does not indicate an error with the device. The protocol STALL condition lasts until the receipt of the next SETUP DP, and the device shall return a STALL TP in response to any IN or OUT transaction on the pipe until the SETUP DP is received. In general, protocol stall indicates that the request or its parameters are not understood by a device and thus provides a mechanism for extending USB requests.

Devices do not support functional stall on a control pipe.

8.12.3 Bus Interval and Service Interval

For all periodic (interrupt and isochronous) endpoints, the interval at which an endpoint must be serviced is called a "Service Interval". In this specification the term "Bus Interval" is used to refer to a 125 µs period.

8.12.4 Interrupt Transactions

The interrupt transfer type is used for infrequent data transfers with a bounded service period. It supports a reliable data transport with guaranteed bounded latency. It offers guaranteed constant data rate as long as data is available. If an error is detected in the data delivered, the host is not required to retry the transaction in the same service interval. However, if a device is momentarily unable to transmit or receive the data (i.e., responds with an NRDY TP), the host shall resume transactions to an endpoint only after it receives an ERDY TP from that device for that endpoint.

Interrupt transactions are very similar to bulk transactions – but are limited to a burst of three DPs in each service interval. The host shall continue to perform transactions to an interrupt endpoint at the agreed upon service interval as long as a device accepts data (in the case of OUT endpoints) or returns data (in the case of IN endpoints). The host is required to send an ACK TP for every DP successfully received in the service interval even if it is the last DP in that service interval. The final ACK TP shall acknowledge the last DP received and shall have the **Number of Packets** field set to zero. If an error occurs while performing transactions to an interrupt endpoint in the current service interval, then the host is not required to retry the transaction in the current service interval but the host shall retry the transaction in the next service interval at the latest.

8.12.4.1 Interrupt IN Transactions

When the host wants to start an Interrupt IN transaction to an endpoint, it sends an ACK TP to the endpoint with the expected sequence number and the number of packets it expects to receive from the endpoint. If an interrupt endpoint is able to send data in response to the ACK TP from the host, it may send up to the number of packets requested by the host within the same service interval. The host shall respond to each of the DPs with an ACK TP indicating successful reception of the data or an ACK TP requesting the DP to be retried in case the DPP was corrupted.

Note that the host expects the first DP to have its sequence number set to zero when it starts the first transfer from a specific endpoint, after the endpoint has been initialized (via a Set Configuration or Set Interface or ClearFeature (ENDPOINT_HALT) command – refer to Chapter 9 for details on these commands).

An interrupt endpoint shall respond to TPs received from the host as described in Section 8.11.1. As long as a device returns data in response to the host sending ACK TPs and the transfer is not complete, the host shall continue to send ACK TPs to the device during every service interval for that endpoint.

The host shall stop performing transactions to an endpoint on the device when any of the following happen:

- The endpoint responds with an NRDY or STALL TP.
- All the data for the transfer is successfully received.
- The endpoint sets the EOB flag in the last DP sent to the host.

When an endpoint receives an ACK TP from the host and cannot respond by sending data, it shall send an NRDY (or STALL in case of an internal endpoint or device error) TP to the host. The host shall not perform any more transactions to the endpoint in subsequent service intervals.

The host shall resume interrupt transactions to an endpoint that responded with a flow control response in a previous service interval only after it receives an ERDY TP from the endpoint. This notifies the host about the endpoint's readiness to transmit data again. Once the host receives the ERDY TP, it shall send an IN request (via an ACK TP) to the endpoint no later than twice the service interval as determined by the value of the **bInterval** field in the interrupt endpoint descriptor. An interrupt endpoint responds by returning either the DP (the sequence number of the packet being one more than the sequence number of the last successful data sent) or, should it be unable to return data, an NRDY or a STALL TP.

If a device receives a deferred interrupt IN TP, and the device needs to send interrupt IN data, the device shall respond with an ERDY TP and keep its link in U0 until it receives the subsequent interrupt transaction from the host, or until tPingTimeout (refer to Table 8-33) time elapses.

As in the case of Bulk transactions, the sequence number is continually incremented with each packet sent by an interrupt endpoint. When the sequence number reaches 31 it wraps around to zero.

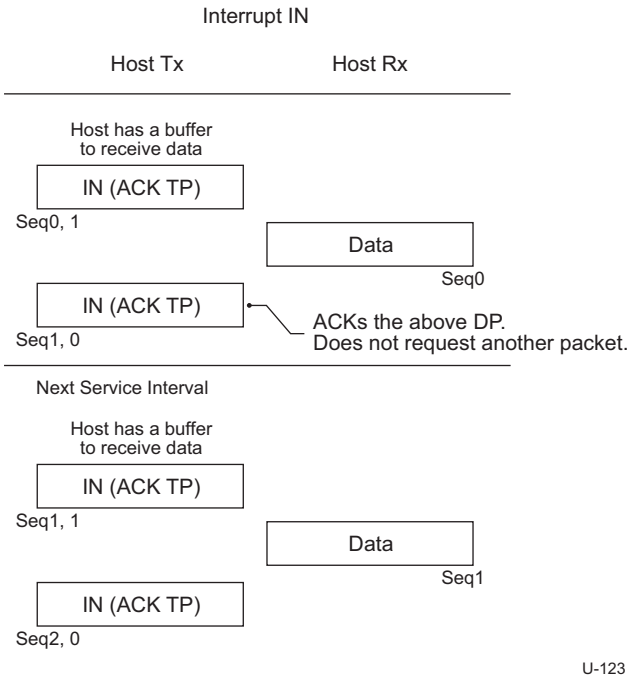


Figure 8-39. Host Sends Interrupt IN Transaction in Each Service Interval

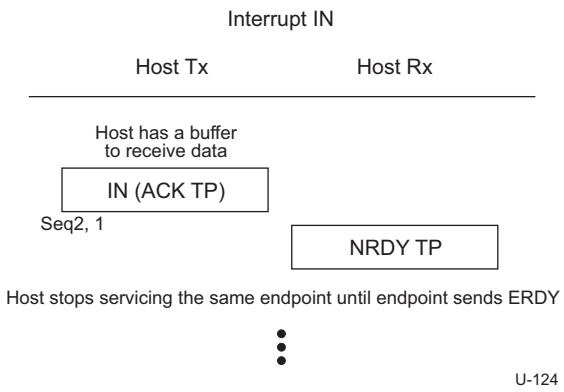


Figure 8-40. Host Stops Servicing Interrupt IN Transaction Once NRDY is Received

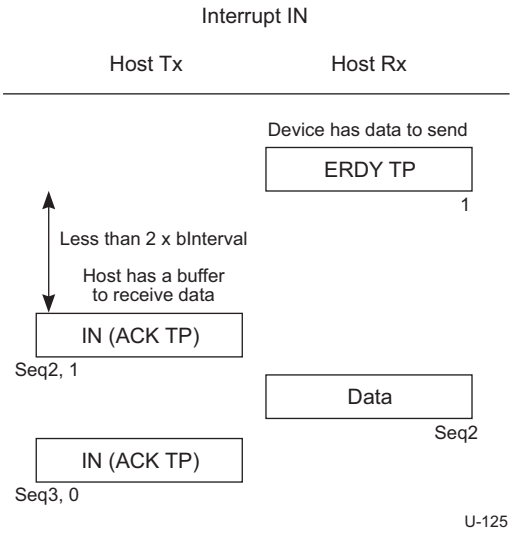


Figure 8-41. Host Resumes IN Transaction after Device Sent ERDY

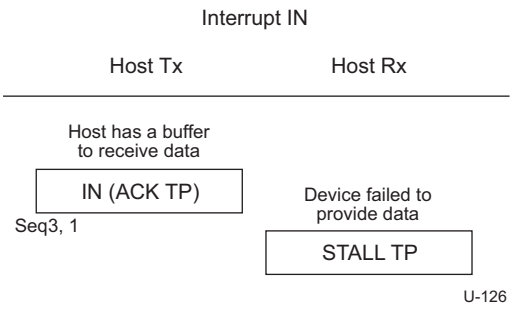


Figure 8-42. Endpoint Sends STALL TP

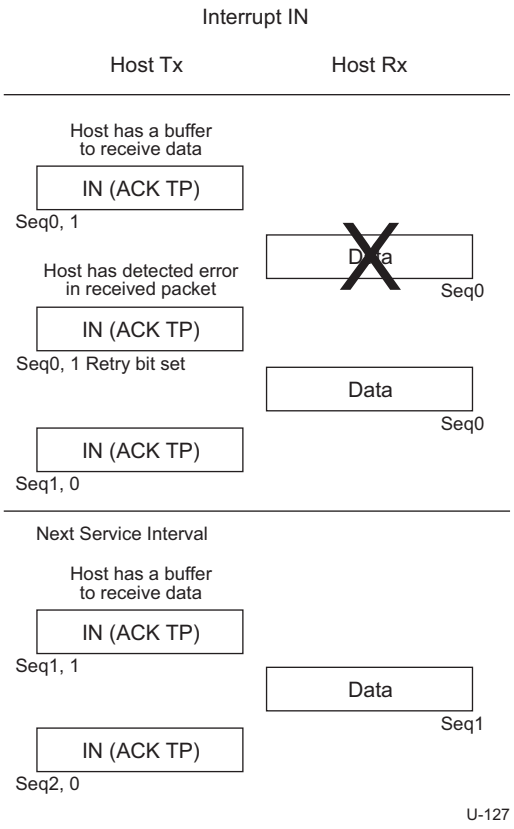


Figure 8-43. Host Detects Error in Data and Device Resends Data

Note: In Figure 8-43 the host retries the data packet received with an error in the same service interval. It is not required to do so and may retry the transaction in the next service interval.

8.12.4.2 Interrupt OUT Transactions

When the host wants to start an Interrupt OUT transaction to an endpoint, it sends the first DP with the expected sequence number. The host may send more packets to the endpoint in the same service interval if the endpoint supports a burst size greater than one. If an endpoint was able to receive that data from the host, it sends an ACK TP to acknowledge the successful receipt of data.

Note that the host always initializes the first DP sequence number to zero in the first transfer it performs to an endpoint after the endpoint is initialized (via a Set Configuration or Set Interface or ClearFeature (ENDPOINT_HALT) command – refer to Chapter 9 for details on these commands).

As long as a device returns ACK TPs in response to the host sending data packets and the transfer is not complete, the host shall continue to send data to the device during every service interval for that endpoint. A device shall acknowledge the successful reception of the DP or ask the host to retry the transaction if the data packet was corrupted.

In response to the OUT data sent by the host an interrupt endpoint shall respond as described in Section 8.11.3.

When an endpoint receives data from the host, and it cannot receive data momentarily, it shall send an NRDY (or STALL in case of an internal endpoint or device error) TP to the host. The host shall not perform any more transactions to the endpoint in subsequent service intervals.

A host shall only resume interrupt transactions to an endpoint that responded with a flow control response after it receives an ERDY TP from that endpoint. This notifies the host about the endpoint's readiness to receive data again. Once the host receives an ERDY TP, the host shall transmit the data packet to the endpoint no later than twice the service interval as determined by the value of the **bInterval** field in the interrupt endpoint descriptor for that endpoint.

If a device receives a deferred interrupt OUT DPH, and the device needs to receive interrupt OUT data, the device shall respond with an ERDY TP and keep its link in U0 until it receives the subsequent interrupt transaction from the host, or until tPingTimeout (see Table 8-33) elapses.

As in the case of Bulk transactions, the sequence number is continually incremented with each packet sent by host. When the sequence number reaches 31 it wraps around to zero.

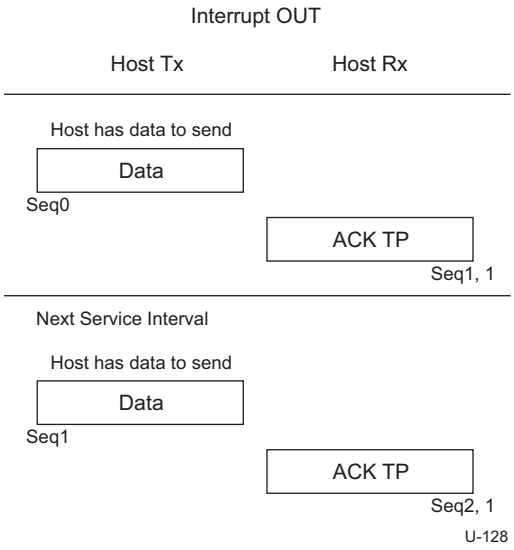


Figure 8-44. Host Sends Interrupt OUT Transaction in Each Service Interval

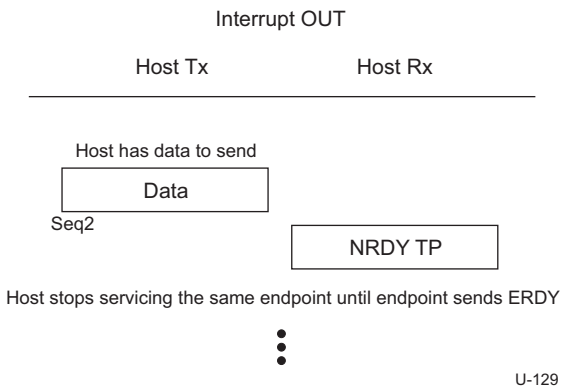


Figure 8-45. Host Stops Servicing Interrupt OUT Transaction Once NRDY is Received

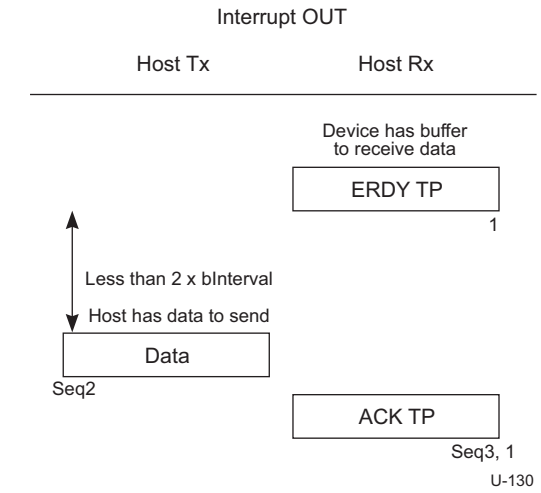


Figure 8-46. Host Resumes Sending Interrupt OUT Transaction After Device Sent ERDY

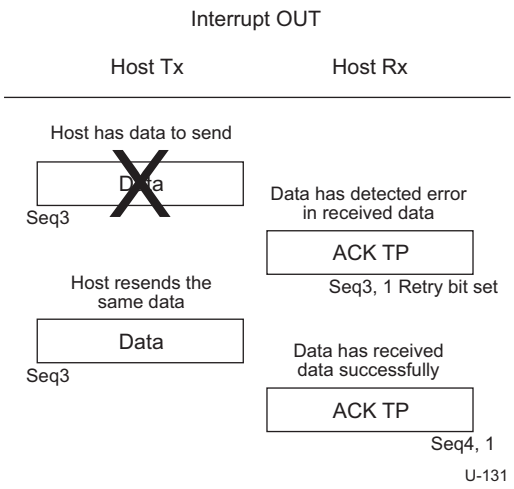


Figure 8-47. Device Detects Error in Data and Host Resends Data

Note: In Figure 8-47 the host retries the data packet received with an error in the same service interval. It is not required to do so and may retry the transaction in the next service interval.

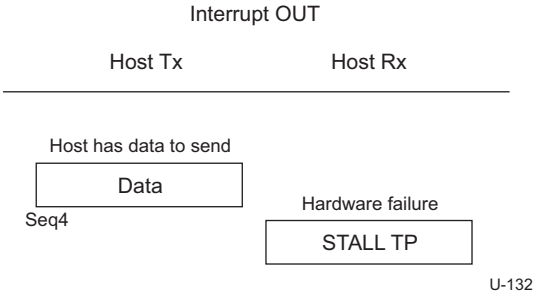


Figure 8-48. Endpoint Sends STALL TP

8.12.5 Host Timing Information

USB 3.0 host controllers do not broadcast regular start of frame (SOF) packets to all devices on a SuperSpeed USB link. Host timing information is only sent by the host via isochronous timestamp packets (ITP) when the root port link is in U0 around a bus interval boundary. Hubs forward isochronous timestamp packets to any downstream port with a link in U0. The host shall provide isochronous timestamps based on a non-spread clock. Devices are responsible for keeping the link in U0 around bus interval boundaries when isochronous timestamps are required for device operation. A device should never keep the link in U0 for the sole purpose of receiving timestamps unless the timestamps are required for device operation.

Note: A device will receive an isochronous timestamp if its link is in U0 around a bus interval boundary. This means that devices without any isochronous endpoints or need for synchronization may discard isochronous timestamp packets without negative side effects.

The timing information is sent in an isochronous timestamp packet around each bus interval boundary and communicates the current bus interval and the time from the start of the timestamp packet to the previous bus interval. Isochronous endpoints request a service interval of $125 * 2^n$ µs, where n is an integer value from 0 to 15 inclusive.

ITPs communicate timing information such that all isochronous endpoints receive the same bus interval boundaries. The host shall keep service interval boundaries aligned for all endpoints at all times unless the host link enters U3. ITPs issued after the host root port link exits U3 may be aligned with boundaries from before the host root port link entered U3. The host shall begin transmitting ITPs within tIsochronousTimestampStart from when the host root port's link enters U0 after the link was in U3. Figure 8-49 shows an example with an active isochronous IN endpoint and isochronous OUT endpoint connected below the same USB 3.0 host controller. The service interval for the isochronous IN endpoint is X and the service interval for the isochronous OUT endpoint is 2X. Note that the host is free to schedule an isochronous IN (via an ACK TP) or isochronous OUT data anywhere within the appropriate service interval. A device will detect the start of new service interval by detecting the rollover of least significant bits in the Bus Interval Counter. The number of bits that need to be monitored for rollover is defined by bInterval. For example, if service interval is equal to two Bus Intervals, the beginning of the service interval is defined by the transition of the least significant bit of the Bus Interval Counter to '0'. If service interval is 4 Bus Intervals, the service interval is defined by the transition of the least significant two bits of the Bus Interval Counter to '0', and so on.

If bInterval is one, a device will detect the start of the service interval when the value of the least significant bit of Bus Interval Counter changes.

A device shall not assume that transactions occur at the same location within each service interval. The host shall schedule isochronous transactions such that they do not cross service interval boundaries.

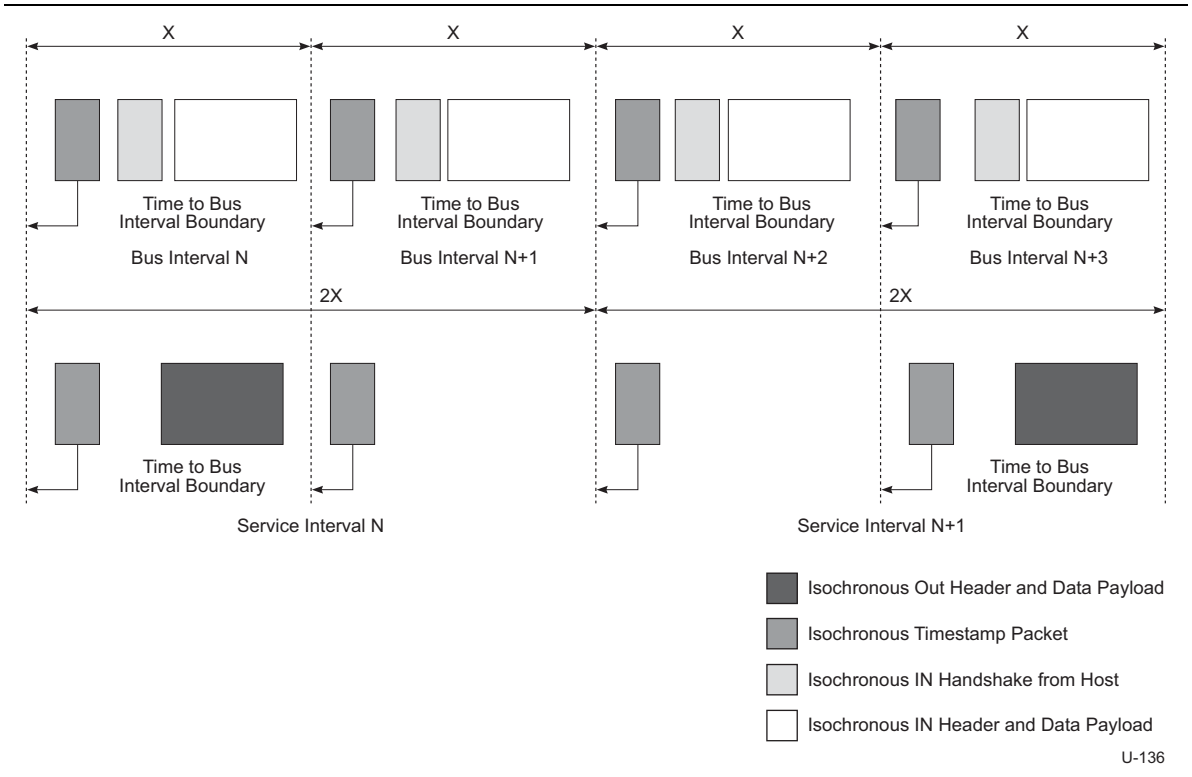


Figure 8-49. Multiple Active Isochronous Endpoints with Aligned Service Interval Boundaries

8.12.6 Isochronous Transactions

IN isochronous transactions are shown in Figure 8-50 and OUT isochronous transactions are shown in Figure 8-51. For INs, the host issues an ACK TP followed by a data phase in which the endpoint transmits data for INs. For OUTs, the host simply transmits data when there is data to be sent in the current service interval. Isochronous transactions do not support retry capability.

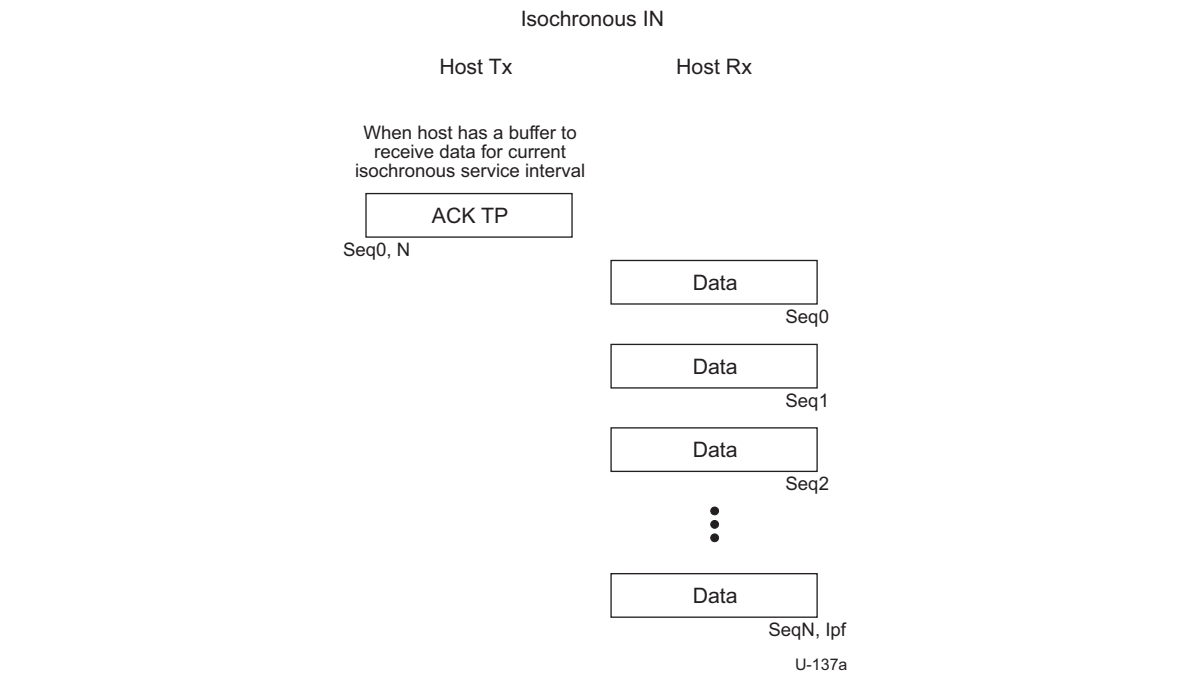


Figure 8-50. Isochronous IN Transaction Format

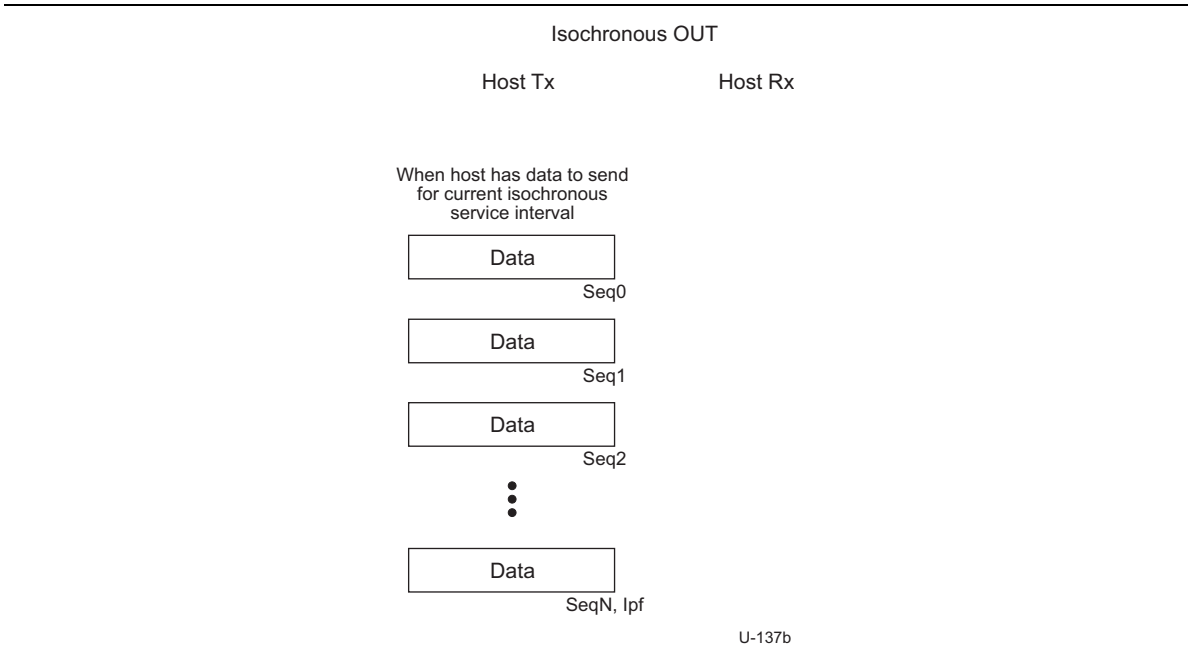


Figure 8-51. Isochronous OUT Transaction Format

SuperSpeed isochronous transactions with a single data packet per service interval always use sequence number zero. For isochronous transactions that include multiple data packets in a service interval the sequence number is increased by one for each subsequent data packet. The first data packet sent in any service interval always uses sequence number zero. The host shall be able to accept and send up to 48 data packets (DP) per service interval. The first DP in each service interval shall start with the sequence number set to 0. The second DP shall have a sequence number set to one; the third DP has a sequence number set to two; and so on until sequence number 31. The next DP after sequence number 31 uses a sequence number of zero. A SuperSpeed device with an isochronous endpoint shall be able to send or receive the number of packets (with sequence numbers 0 – N) as indicated in its endpoint and endpoint companion descriptors.

The last packet in the service interval shall be sent with the lpf field set to 1 and can be less than or equal to MaxPacketSize bytes. Each packet except the last packet in the service interval shall be sent with the lpf field set to 0 and shall be equal to MaxPacketSize bytes. If there is no data to send to an isochronous OUT endpoint during a service interval, the host does not send anything during the interval. If a device with an isochronous IN endpoint does not have data to send when an isochronous IN ACK TP is received from the host, it shall send a zero length data packet.

Figure 8-52 and Figure 8-53 show sample isochronous IN and OUT transactions for endpoints that have requested 2000 bytes of bandwidth per service interval (i.e., no more than two packets can be sent or received each service interval).

If the host is not able to send isochronous OUT data during the specified interval due to an error condition, the host discards the data and notifies host software of the error. If the host is not able to send an isochronous ACK TP during the specified service interval due to an error condition, the host notifies host software of the error.

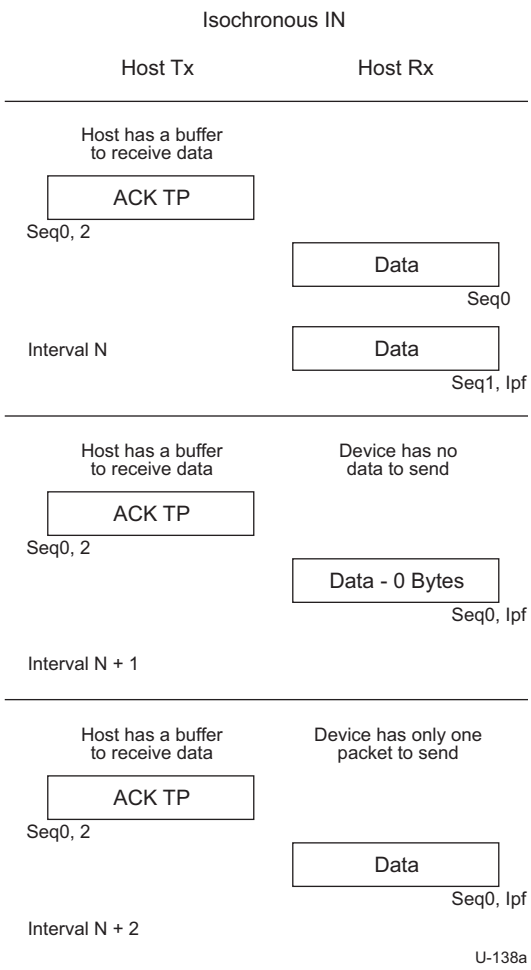


Figure 8-52. Sample Isochronous IN Transaction

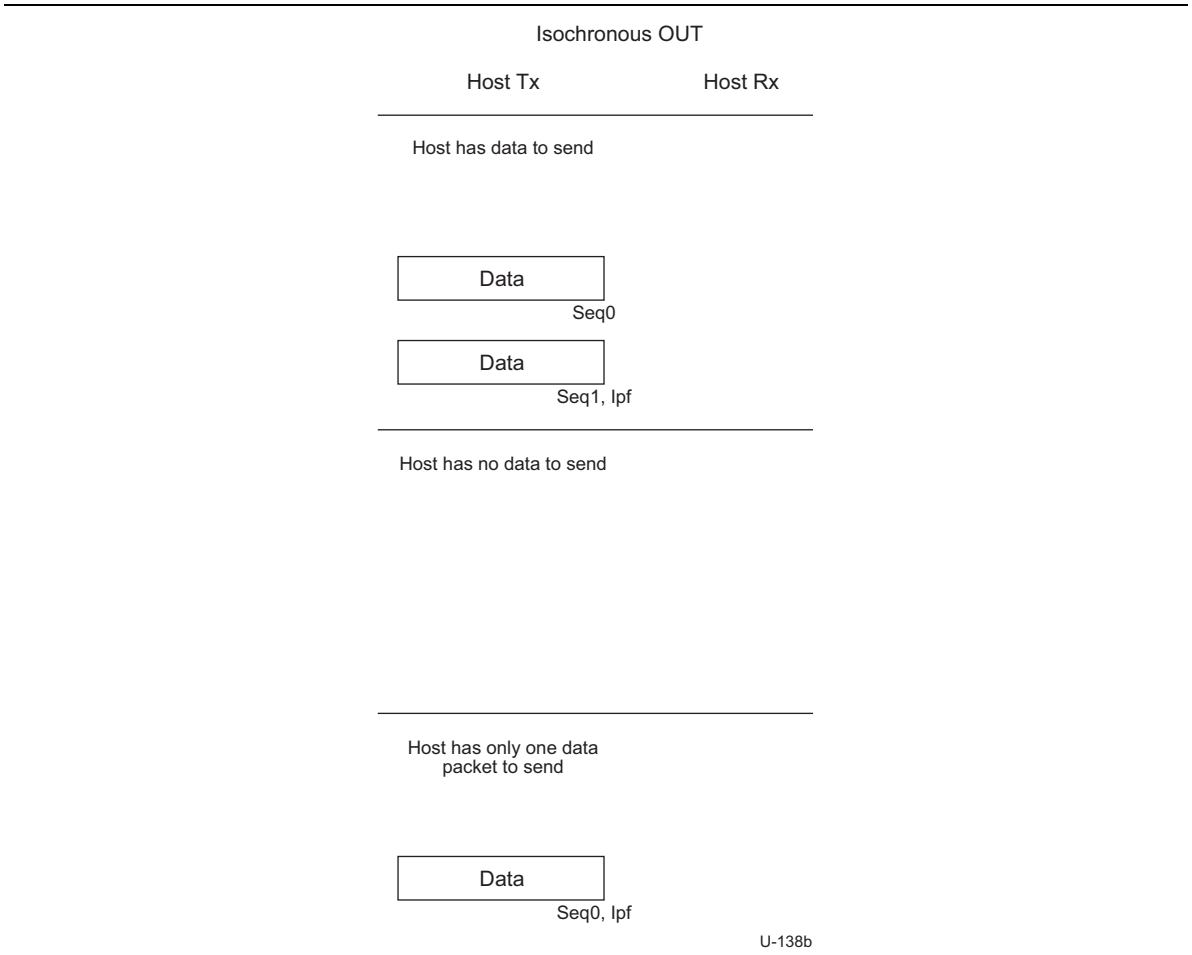


Figure 8-53. Sample Isochronous OUT Transaction

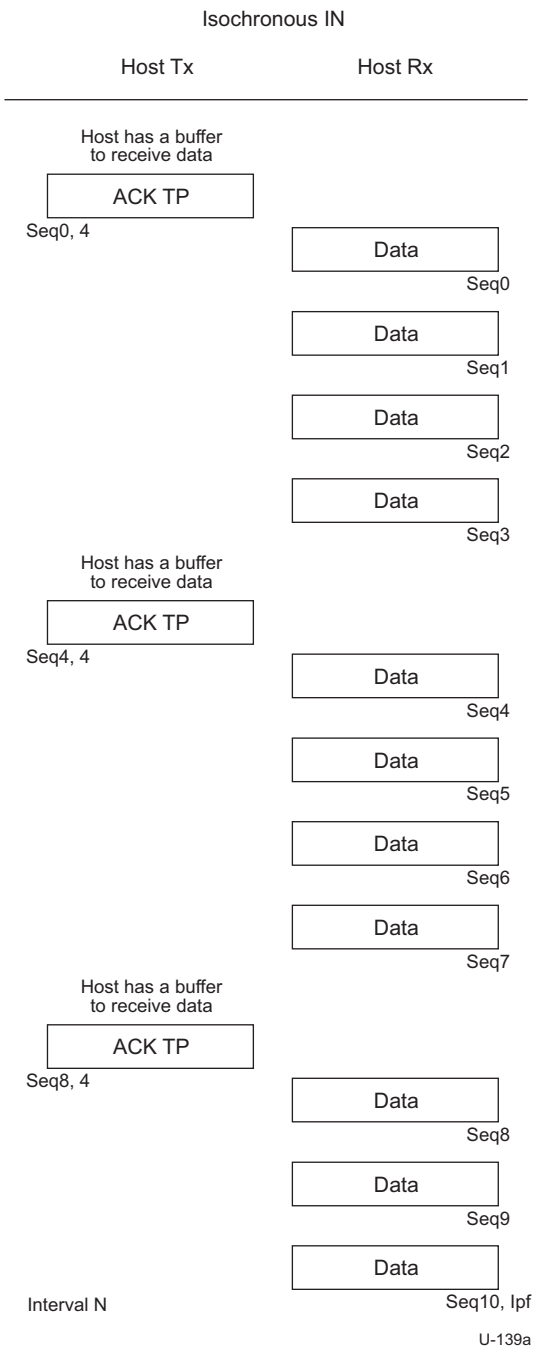


Figure 8-54. Isochronous IN Transaction Example

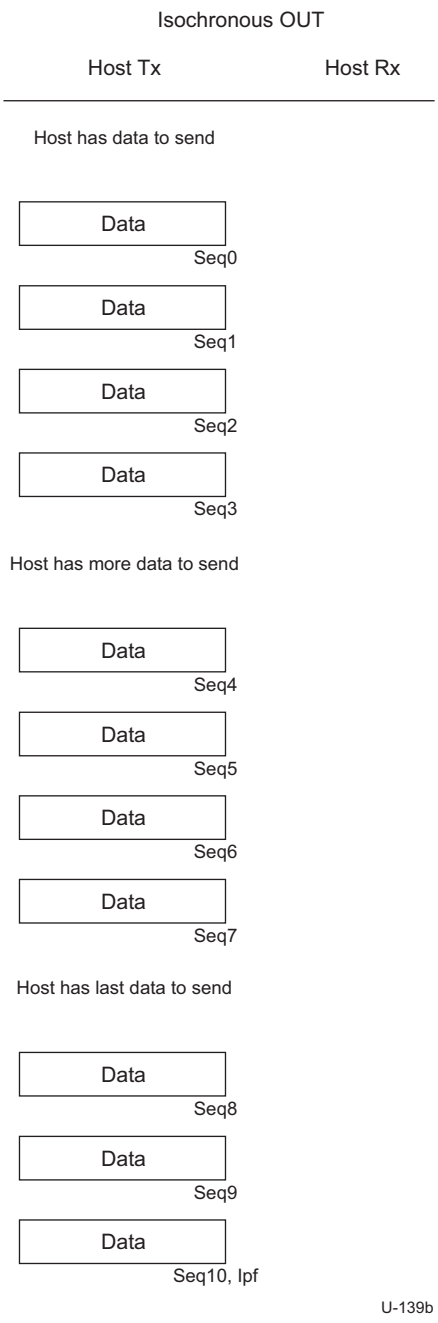
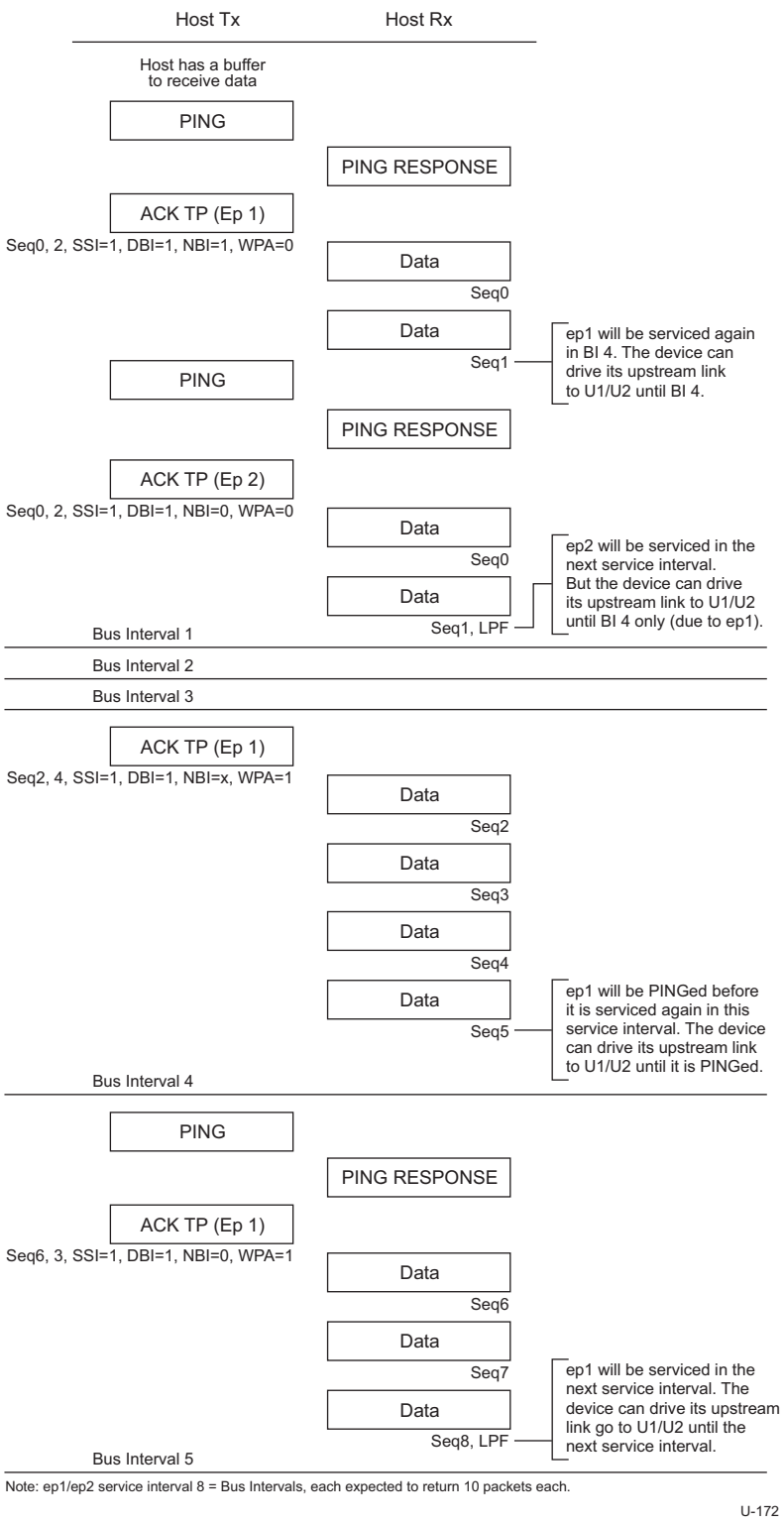


Figure 8-55. Isochronous OUT Transaction Example

Figure 8-56 and Figure 8-57 show sample isochronous IN and OUT transactions with smart Isochronous scheduling to endpoints that have service intervals of 8. In the isochronous IN example below the host is only sending one ACK TP with the SSI and DBI field set to non-zero values when asking for data from the endpoint. It should be noted that a host may send multiple ACK TPs with only the last ACK TP in the current bus interval having the SSI and DBI fields set to non-zero values.



Note: ep1/ep2 service interval 8 = Bus Intervals, each expected to return 10 packets each.

U-172

Figure 8-56. Example Smart Isochronous IN Transaction

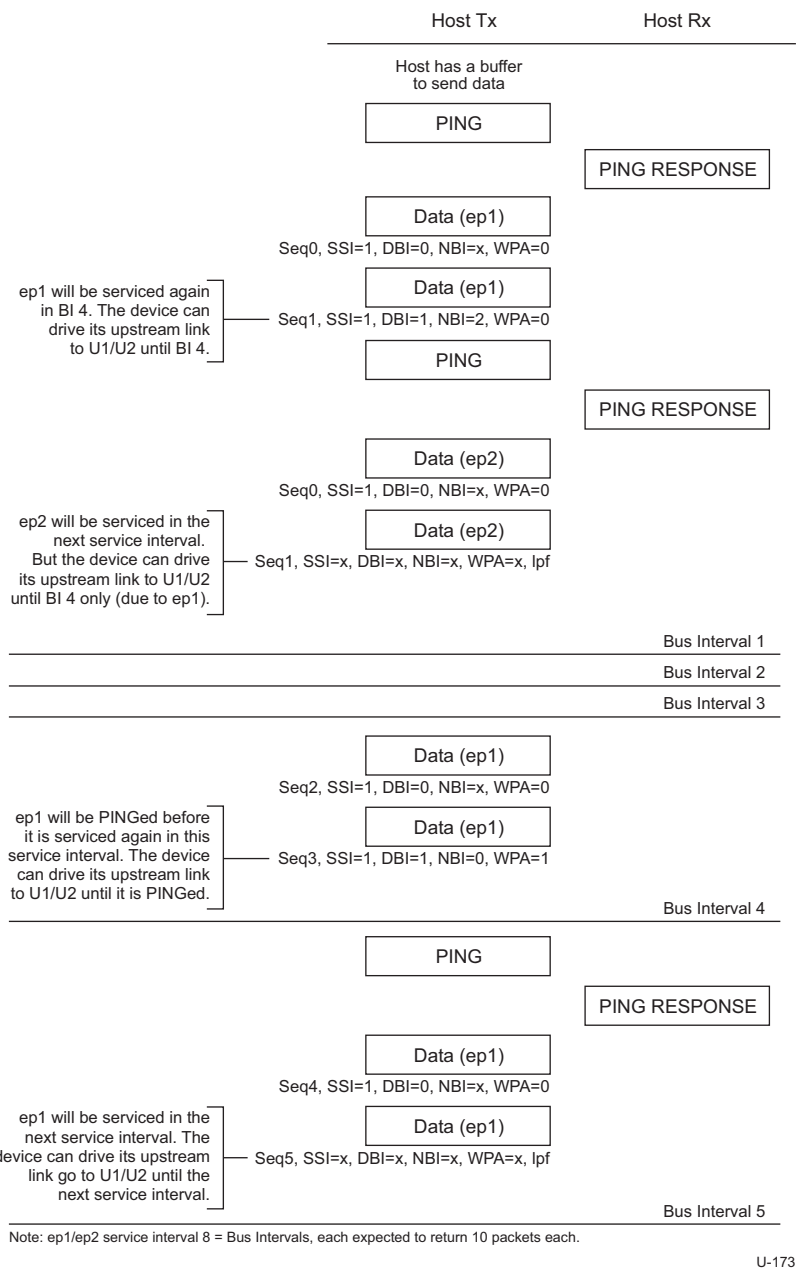


Figure 8-57. Example Smart Isochronous OUT Transaction

8.12.6.1 Host Flexibility in Performing Isochronous Transactions

The host is allowed some flexibility in performing isochronous service during a service interval. The host may transfer all the DPs to or from an endpoint as a single isochronous burst transaction or it may split the transfer into smaller bursts of two, four, or eight DPs followed by a final isochronous burst with the remaining DPs for that service interval. The host shall not perform isochronous transactions in any other way. For isochronous endpoints that have a multiplier value greater than one, these rules apply to the burst transactions associated with each multiplier value separately. A device shall support all possible host burst transactions allowed by these rules. For example, if an isochronous IN endpoint requests a maximum number of packets in a burst of 11 and the host has 11 packets to receive from the endpoint during a service interval there are four possible ways the host could perform the transaction:

- Request a single burst of 11 packets
- Request a burst of eight followed by a burst of three
- Request two bursts of four followed by a burst of three
- Request five bursts of two followed by a burst of one
- Request 11 bursts of one.

Taking the above example a step further, if the isochronous IN endpoint requests a maximum number of packets in a burst of 11 and a Mult of 2 (in essence requesting three bursts of 11) and the host has buffer space to receive 33 packets from the endpoint during a service interval, then the host can use any combination of the above mentioned options to transfer the three sets of 11 packets to the endpoint.

The **SSI**, **WPA**, **DBI** and **NBI** fields (described in Table 8-12) are provided in addition to the lpf to give devices more information about when the host plans to transfer isochronous data thus allowing them to more aggressively manage their upstream link. The DBI is used to tell the device that the host has no more data to transfer during the current bus interval. The WPA field, when set to one, informs the device that the host will send a PING TP to the device before it initiates a data transfer on the endpoint again.

The **NBI** value provides the device additional information (when **DBI** is set to one and **WPA** is set to zero) that it may use to more aggressively manage its upstream port. The value is used to determine the bus interval number (see Table 8-12) that the host will initiate another data transfer on the endpoint. In this case, the host will not be required to send a PING TP before it resumes transfers to the endpoint; it is the device's responsibility to manage its upstream port's link accordingly.

Note that the SSI and related fields are only valid and may only be used by a host to inform a device about the manner in which it will service a particular isochronous endpoint on a device within a service interval. A host is always required to send a PING and wait for a PING_RESPONSE before servicing an isochronous endpoint before the start of each service interval.

8.12.6.2 Device Response to Isochronous IN Transactions

Table 8-30 lists the possible responses a device may make in response to an ACK TP. An ACK TP is considered to be invalid if it has an incorrect Device Address or the endpoint number and

direction does not refer to an endpoint that is part of the current configuration or it does not have the expected sequence number or it has the deferred bit set in it.

Table 8-30. Device Responses to Isochronous IN Transactions

ACK TP Received Invalid	Device Can Transmit Data	Action Taken
Yes	Do not care	Return no response
No	No	Return zero length data packet with sequence number 0
No	Yes	Return N data packets with sequence numbers 0 to N-1. Each packet except the last shall be MaxPacketSize bytes. The last packet can be less than or equal to MaxPacketSize bytes. The last packet shall have the LPF flag set.

8.12.6.3 Host Processing of Isochronous IN Transactions

Table 8-31 lists the host processing of data from an IN transaction. The host never returns a response to isochronous IN data received. In Table 8-31, DP Error may be due to one or more of the following:

- CRC-32 incorrect
- DPP aborted
- DPP missing
- Data length in the DPH does not match the actual data payload length.

If the host receives a corrupted data packet, it discards the remaining data in the current service interval and informs host software of the error.

Table 8-31. Host Responses to IN Transactions

Data Packet Error	Host Can Accept Data	Host Data Processing
Yes	N/A	Discard data
No	No. (This should never happen for a compliant host implementation.)	Discard data
No – Data Packet Has Expected Sequence Number	Yes	Accept data
No – Data Packet Does Not Have Expected Sequence Number.	Yes	Discard data

8.12.6.4 Device Response to an Isochronous OUT Data Packet

Table 8-32 lists the device processing of data from an OUT data packet. A device never returns a TP in response. In Table 8-32, DP Error may be due to one or more of the following:

- CRC-32 incorrect
- DPP aborted
- DPP missing
- Data length in the DPH does not match the actual data payload length
- Deferred bit set in the DPH

Table 8-32. Device Responses to OUT Data Packets

Data Packet Error	Expected Sequence Number	Device Can Accept Data	Device Data Processing
Yes	Do not care	Do not care	Discard data
No	Yes	Yes	Accept data
No	Yes	No	Data discarded
No	No	No	Data discarded. Device may discard any additional data for current service interval.
No	No	Yes	Data discarded. Device may discard any additional data for current service interval.

8.13 Timing Parameters

Table 8-33 lists the minimum and/or maximum times a device shall adhere to when responding to various types of packets it receives. It also lists the default and minimum times a device may set in Latency Tolerance messages as well as the minimum time after receipt of certain TPs and when it can initiate a U1 or U2 entry. In addition, it lists the maximum time between DPs a device must adhere to while bursting.

Note that all txxxResponse (e.g., tNRDYResponse) and tMaxBurstInterval times are all timings that a device shall meet when the device has nothing else to send on its upstream link.

Table 8-33. Timing Parameters

Name	Description	Min	Max	Units
tPortConfiguration	Maximum duration from when the port enters U0 to when it has completed the exchange of LMPs. In case of tiebreaker (refer to Table 8-7), both ports shall reset their timers and restart them for each LMP exchange until the tiebreaker is resolved.		20	µs
tPingTimeout	Timeout after a device receives a ping from the host and when it can initiate or accept U1 or U2 requests. This parameter is measured in terms of the maximum of all the service intervals for all isochronous endpoints within the device.	2		Service intervals
tPingResponse	Time between device reception of the last framing symbol of a ping and the first framing symbol of the ping_response		400	ns
tBELTDefault	Default for best effort latency tolerance	1		ms
tBELTmin	Minimum value of best effort latency tolerance allowed in a Latency Tolerance Message	125		µs
tNRDYorSTALLResponse	Time between device reception of the last framing symbol for an ACK TP or a DPP or a STATUS TP and the first framing symbol of an NRDY or STALL response		400	ns
tDPResponse	Time between device reception of the last framing symbol for an ACK TP and the first framing symbol of a DP response		400	ns
tACKResponse	Time between device reception of the last framing symbol for a DPP or a STATUS TP and the first framing symbol of an ACK response		400	ns
tHostACKResponse	Time between host reception of the last framing symbol for a DPP and the first framing symbol of an ACK response		3	µs
tERDYTimeout	Timeout after a device sends an ERDY to the host and when it can initiate or accept a U1 or U2 request if not serviced	500		ms
TNotification	Rate at which a device shall send a function wake notification if the device has not been accessed (since sending the last function wake notification)	2500		ms

Name	Description	Min	Max	Units
tMaxBurstInterval	Time between DPs when a device or host is bursting. If the device cannot meet this maximum time, then it should set the EOB flag in the last DP it sends.		100	ns
tTimestampWindow	The host shall transmit an isochronous timestamp from a bus interval boundary to tTimestampWindow after the bus interval boundary if the root port's link is in U0.	0	8	µs
tIsochTimestampGranularity	The granularity of isochronous timestamps	8	8	USB 2.0 High-Speed bit times
BusIntervalAdjustmentGranularity	The adjustment unit for device requested changes to the bus interval		4.0690104[1]	ps
tIsochronousTimestampStart	Time by which the host shall start transmitting isochronous timestamps after a root port link enters U0 from polling or after the root port link enters U0 after the link was in U3		250	µs
tBELTRepeat	Duration within which devices are limited to send more than two LTM TPs	1		ms
tMinLTMStateChange	Time by which a peripheral device must send an LTM notification after completion of request to enable or disable LTM_Enable feature selector		10	µs
tHostTransactionTimeout	For control, bulk, and interrupt transactions, this is defined as the time without receiving a response to the last DP or ACK TP that the host sent out before the host shall assume that the transaction has failed and halt the endpoint. For Isochronous IN transactions, this is defined as the time without receiving a response to the ACK TP that a host sent. The timer is initialized and restarts counting whenever the host receives each DP that was requested by the ACK TP. If a timeout occurs, the host shall not perform any more transactions to the endpoint in the current service interval. The host shall not halt the endpoint and shall restart transactions to the endpoint in the next service interval. No retries shall be performed.	32	5032	µs

[1] (tIsochTimestampGranularity/4096)

9 Device Framework

A device may be divided into three layers:

- The bottom layer is a bus interface that transmits and receives packets.
- The middle layer handles routing data between the bus interface and various endpoints on the device. As in USB 2.0, the endpoint is the ultimate consumer or provider of data. It may be thought of as a source or sink for data. The characteristics of an endpoint; e.g., the endpoint's transfer type, the maximum payload (MaxPacketSize), and the number of packets (Burst Size) it can receive or send at a time are described in the endpoint's descriptor.
- The top layer is the functionality provided by the serial bus device, for instance, a mouse or video camera interface.

This chapter describes the common attributes and operations of the middle layer of a device. These attributes and operations are used by the function-specific portions of the device to communicate through the bus interface and ultimately with the host.

9.1 USB Device States

A device has several possible states. Some of these states are visible to the USB and the host, while others are internal to the device. This section describes those states.

9.1.1 Visible Device States

This section describes device states that are externally visible (see Figure 9-1). Table 9-1 summarizes the visible device states.

NOTE

Devices perform a reset operation in response to reset signaling on the upstream facing port. When reset signaling has completed, the device is reset. The reset signaling depends on the link state. Refer to Section 7.3 for details.

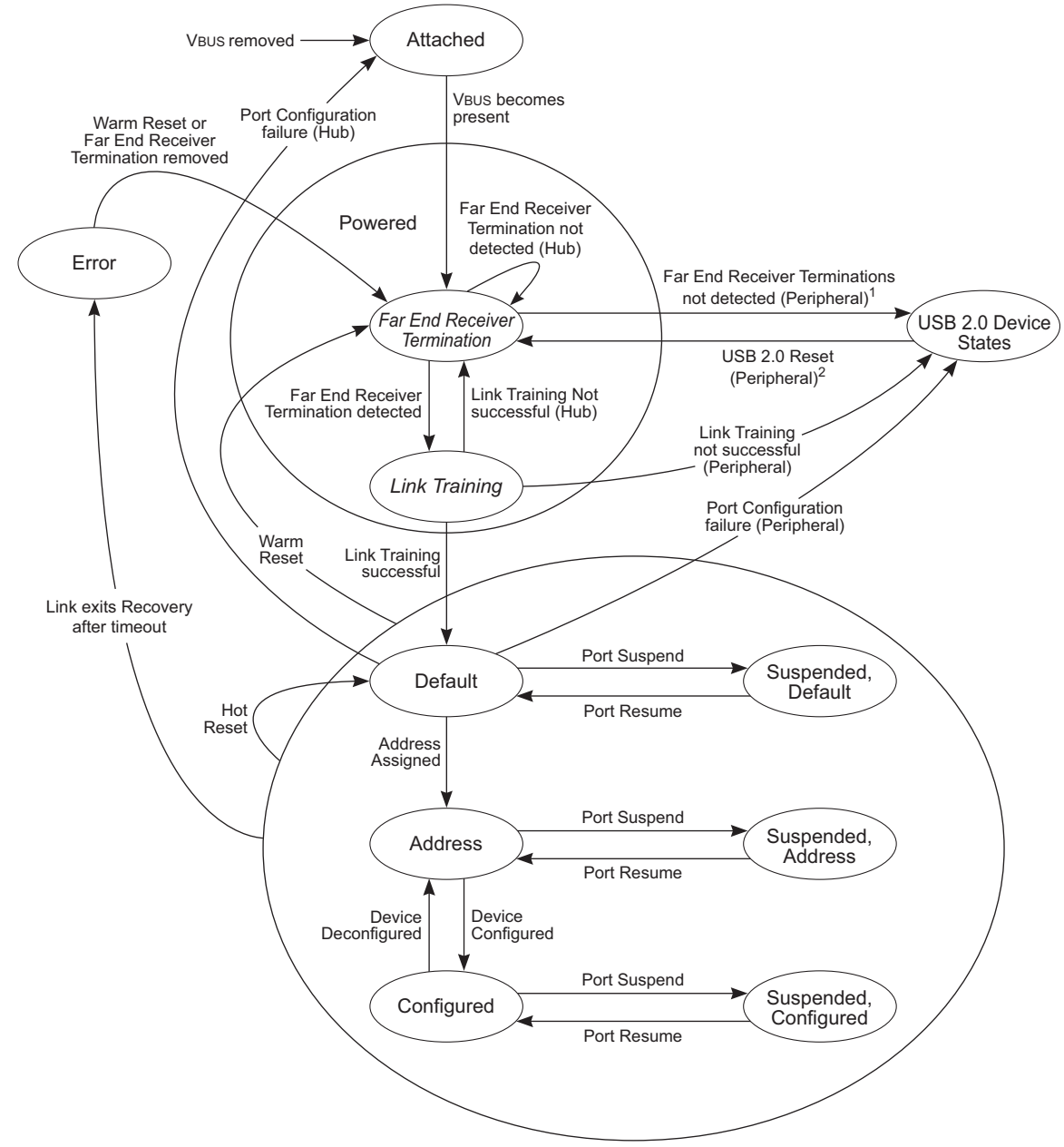


Figure 9-1. Peripheral State Diagram and Hub State Diagram (SuperSpeed Portion Only)

Figure 9-1 is a combined state diagram for both peripherals and hubs. Note that a USB 3.0 Hub has two discrete state diagrams, one for the SuperSpeed portion shown in Figure 9-1**Error! Reference source not found.** and another for the non-SuperSpeed portion which may be found in Figure 9-1 in the *USB 2.0 Specification.*

Table 9-1. Visible SuperSpeed Device States

Attached	Powered	Default	Address	Configured	Suspended[1]	Error	State
No	--	--	--	--	--	--	Device is not attached to the USB. Other attributes are not significant.
Yes	No	--	--	--	--	--	Device is attached to the USB, but is not powered. Other attributes are not significant.
Yes	Yes	No	--	--	--	--	Device is attached to the USB and powered and its upstream link has not successfully completed training.
Yes	Yes	Yes	No	--	--	--	Device is attached to the USB and powered and has been reset, but has not been assigned a unique address. Device responds at the default address.
Yes	Yes	Yes	Yes	No	--	--	Device is attached to the USB, powered, has been reset, and a unique device address has been assigned. Device is not configured.
Yes	Yes	Yes	Yes	Yes	No	--	Device is attached to the USB, powered, has been reset, has a unique address, is configured, and is not suspended. The host may now use the function provided by the device.
Yes	Yes	Yes	--	--	Yes	--	Device is, at minimum, in the Default State (attached to the USB, is powered and its upstream link has been successfully trained) and its upstream link has been set to U3 by its upstream link partner. It may also have a unique address and be configured for use. However, because the device is suspended, the host may not use the device's function.
Yes	Yes					Yes	Device is attached to the USB, powered, and a link timeout error has occurred.

[1]Suspended from the Default, Address, or Configured state.

9.1.1.1 Attached

A device may be attached or detached from the USB. The state of a device when it is detached from the USB is not defined by this specification. This specification only addresses required operations and attributes once the device is attached.

9.1.1.2 Powered

Devices may obtain power from an external source and/or from the USB through the hub to which they are attached. Externally powered devices are termed self-powered. Although self-powered devices may already be powered before they are attached to the USB, they are not considered to be in the Powered state until they are attached to the USB and VBUS is applied to the device.

A device may support both self-powered and bus-powered configurations. Some device configurations support either power source. Other device configurations may be available only if the device is self-powered. Devices report their power source capability through the configuration descriptor. The current power source is reported as part of a device's status. Devices may change their power source at any time, e.g., from self- to bus-powered. If a configuration is capable of supporting both power modes, the power maximum reported for that configuration is the maximum the device will draw from VBUS in either mode. The device shall observe this maximum, regardless of its mode. If a configuration supports only one power mode and the power source of the device changes, the device will lose its current configuration and address and return to the Powered state. If a device operating in SuperSpeed mode is self-powered and its current configuration requires more than 150 mA, then if the device switches to being bus-powered, it shall return to the Powered state. Self-powered hubs that use VBUS to power the Hub Controller are allowed to remain in the Configured state if local power is lost. Note that the maximum power draw for a SuperSpeed device operating in non-SuperSpeed mode is governed by the limits set in the USB 2.0 specification.

A hub port shall be powered in order to detect port status changes, including attach and detach. Bus-powered hubs do not provide any downstream power until they are configured, at which point they will provide power as allowed by their configuration and power source. A device shall be able to be addressed within a specified time period from when power is initially applied (refer to Chapter 7). After an attachment to a port has been detected, the host may reset the port, which will also reset the device attached to the port.

While in the Powered state, a hub or peripheral device may be in one of two substates: "Far-end Receiver Termination" or "Link Training".

9.1.1.2.1 Far-end Receiver Termination Substate

A peripheral device shall transition to USB 2.0 Device States as per the conditions defined in Note 2 of Figure 10-25 if Far-end Receiver Terminations are not detected.

A hub shall remain in the Far-end Receiver Termination substate if Far-end Receiver Terminations are not detected.

If Far-end Receiver Terminations are detected, a hub or peripheral device shall transition to the Link Training substate.

9.1.1.2.2 Link Training Substate

A peripheral device shall transition to USB 2.0 Device States if Link Training fails.

A hub shall transition to the Far-end Receiver Termination substate if Link Training fails.

If Link Training is successful, a hub or peripheral device shall transition to the Default state.

9.1.1.3 Default

When operating in SuperSpeed mode, after the device has been powered, it shall not respond to any bus transactions until its link has successfully trained. The device is then addressable at the default address.

A device that is capable of SuperSpeed operation determines whether it will operate at SuperSpeed as a part of the connection process (see the Device Connection State Diagram in Chapter 10 for more details).

A USB 3.0 device shall reset successfully at one of the supported USB 2.0 speeds when in an USB 2.0 only electrical environment. After the device is successfully reset, the device shall also respond successfully to device and configuration descriptor requests and return appropriate information according to the requirements laid out in the USB 2.0 specification. The device may or may not be able to support its intended functionality when operating in the USB 2.0 mode.

A peripheral device shall transition to USB 2.0 Device States, if Port Configuration fails. Refer to Section 8.4.5.

A hub shall transition to the Attached state if Port Configuration fails. Note that it is necessary to physically remove and reapply Vbus to transition a hub out of the Attached state.

9.1.1.4 Address

All devices use the default address when initially powered or after the device has been reset. Each device is assigned a unique address by the host after reset. A device maintains its assigned address while suspended.

A device responds to requests on its default pipe whether the device is currently assigned a unique address or is using the default address.

9.1.1.5 Configured

Before a device's function may be used, the device shall be configured. From the device's perspective, configuration involves correctly processing a SetConfiguration() request with a non-zero configuration value. Configuring a device or changing an alternate setting causes all of the status and configuration values associated with all the endpoints in the affected interfaces to be set to their default values. This includes resetting the sequence numbers of any endpoint in the affected interfaces to zero. On initial entry into the configured state a device shall default into the fully functional D0 State.

9.1.1.6 Suspended

In order to conserve power, devices automatically enter the Suspended state (one of Suspended Default, Address, or Configured) when they observe that their upstream link is being driven to the U3 state (refer to Section 7.2.4.2.4). Refer to Section 9.2.5.2 for the state that a device maintains while it is suspended.

Attached devices shall be prepared to suspend at any time from the Default, Address, or Configured states. A device shall enter the Suspended state when the hub port it is attached to is set to go into U3. This is referred to as selective suspend.

A device exits suspend mode when it observes wake-up signaling (refer to Section 6.9.1 and Section 7.5.9) on its upstream port. A device may also request the host to exit suspend mode or selective suspend by driving resume signaling (refer to Section 6.9.1 and Section 7.5.9) and sending a Function Wake Notification (refer to Section 8.5.6) on its upstream link to indicate remote wakeup. The ability of a device to signal remote wakeup is optional. If a device is capable of remote wakeup, the device shall support the ability of the host to enable and disable this capability. When the device is reset, remote wakeup shall be disabled. Refer to Section 9.2.5 for more information.

9.1.1.7 Error

This state is entered if the device is in the Default, Address, Configured, or Suspended state and its link exits the Recovery state due to a timeout. A Warm Reset or removal of Far-end Receiver Terminations shall recover from this error condition and transition the device to the Powered:Far-end Receiver Termination substate.

9.1.2 Bus Enumeration

When a device is attached to or removed from the USB, the host uses a process known as bus enumeration to identify and manage the device state changes necessary. When a device is attached to a powered port, the following actions are taken (note, these actions apply whether the attached device is a peripheral device or hub device):

1. The hub to which the device is now attached informs the host of the event via a reply on its status change pipe (refer to Section 10.11.1). At this point, the device has been reset, is in the Default state and the port to which it is attached is enabled and ready to respond to control transfer requests on the default control pipe.
2. The host determines the exact nature of the change by querying the hub.
3. Now that the host knows the port to which the new device has been attached, the host then may reset the device again if it wishes but it is not required to do so.
4. If the host resets the port, the hub performs the required reset processing for that port (refer to Section 10.3.1.6). When the reset is completed, the port will be back in the enabled state.
5. The device is now in the Default state and can draw no more than 150 mA from V_{BUS}. All of its registers and state have been reset and it answers to the default address.
6. The host assigns a unique address to the device, moving the device to the Address state.
7. Before the device receives a unique address, its default control pipe is still accessible via the default address. The host reads the device descriptor to determine what the actual maximum data payload size this device's default pipe can use.

8. The host shall set the isochronous delay to inform the device of the delay from the time a host transmits a packet to the time it is received by the device.
9. The host shall inform the device of the system exit latency using the Set SEL request. Device shall accept a Set SEL request whether it is LTM capable or not and whether LTM is enabled or not.
10. The host reads the configuration information from the device by reading each configuration from zero to *n*-1, where *n* is the number of configurations. This process may take several milliseconds to complete.
11. Any time after this, the host can set the U1/U2 timeout for the downstream port on which the device is connected using the Set Port Feature (PORT_U1_TIMEOUT/PORT_U2_TIMEOUT).
12. Based on the configuration information and how the device will be used, the host assigns a configuration value to the device. The device is now in the Configured state and all of the endpoints in this configuration have taken on their described characteristics. The device may now draw the amount of VBUS power described in its descriptor for the selected configuration. From the device's point of view, it is now ready for use.

When the device is detached, the hub again sends a notification to the host. Detaching a device disables the port to which it had been attached and the port moves into the Disconnected state (refer to Section 10.3.1.2). Upon receiving the detach notification, the host will update its local topological information.

9.2 Generic Device Operations

All devices support a common set of operations. This section describes those operations.

9.2.1 Dynamic Attachment and Removal

Devices may be attached or detached at any time. The hub provides the attachment point or downstream port and is responsible for reporting any change in the state of the port.

The hub resets and enables the hub downstream port where the device is attached upon detection of an attachment, which also has the effect of resetting the device. A reset device has the following characteristics:

- Its USB address is set to zero (the default USB address)
- It is not configured
- It is not suspended

When a device is removed from a hub port, the hub disables the port where the device was attached, the port moves into the DSPORT.Disconnected state (refer to Section 10.3.1.2) and notifies the host of the removal.

9.2.2 Address Assignment

When a device is attached, the host is responsible for assigning a unique address to the device. Before assigning an address, the host may explicitly reset the device; however, note that the device implicitly gets reset during the connection process before the host is notified of a device being attached to the port.

9.2.3 Configuration

A device shall be configured before its function(s) may be used. The host is responsible for configuring a device. The host typically requests configuration information from the device to determine its capabilities.

As part of the configuration process, the host sets the device configuration and, where necessary, selects the appropriate alternate settings for the interfaces.

Within a single configuration, a device may support multiple interfaces. An interface is a related set of endpoints that present a single feature or function of the device to the host. The protocol used to communicate with this related set of endpoints and the purpose of each endpoint within the interface may be specified as part of a device class or vendor-specific definition.

In addition, an interface within a configuration may have alternate settings that redefine the number or characteristics of the associated endpoints. If this is the case, the device shall support the GetInterface() request to report the current alternate setting for the specified interface and SetInterface() request to select the alternate setting for the specified interface.

Within each configuration, each interface descriptor contains fields that identify the interface number and the alternate setting. Interfaces are numbered from zero to one less than the number of concurrent interfaces supported by the configuration. Alternate settings range from zero to one less than the number of alternate settings for a specific interface. The default setting when a device is initially configured is alternate setting zero.

In support of adaptive device drivers that are capable of managing a related group of devices, the device and interface descriptors contain *Class*, *SubClass*, and *Protocol* fields. These fields are used to identify the function(s) provided by a device and the protocols used to communicate with the function(s) on the device. A class code is assigned to a group of related devices that has been characterized as a part of a USB Class Specification. A class of devices may be further subdivided into subclasses and, within a class or subclass, a protocol code may define how the host software communicates with the device.

Note: The assignment of class, subclass, and protocol codes shall be coordinated but is beyond the scope of this specification.

9.2.4 Data Transfer

Data may be transferred between an endpoint within a device and the host in one of four ways. Refer to Chapter 4 for the definition of the four types of transfers. An endpoint number may be used for different types of data transfers in different alternate settings. However, once an alternate setting is selected (including the default setting of an interface), a device endpoint uses only one data transfer method until a different alternate setting is selected.

9.2.5 Power Management

Power management on devices involves the issues described in the following sections.

9.2.5.1 Power Budgeting

USB bus power is a limited resource. During device enumeration, a host evaluates a device's power requirements. If the power requirements of a particular configuration exceed the power available to the device, host software shall not select that configuration.

Devices shall limit the power they consume from VBUS to one unit load or less until configured. When operating in SuperSpeed mode, 150 mA equals one unit load. Suspended devices, whether configured or not, shall limit their bus power consumption as to the suspend mode power requirements in the USB 2.0 specification. Depending on the power capabilities of the port to which the device is attached, a SuperSpeed device operating in SuperSpeed mode may be able to draw up to six unit loads from VBUS after configuration. The amount of current draw for SuperSpeed devices are increased to 150 mA for low-power devices and 900 mA for high-power devices when operating in SuperSpeed mode.

Device power management is comprised of suspend and function suspend. Suspend refers to a device-wide state that is entered when its upstream link is placed in U3. Function suspend refers to a state of an individual function within a device. Suspending a device with more than one function effectively suspends all the functions within the device.

Note that placing all functions in the device into function suspend does not suspend the device. A device is suspended only when its upstream link is placed in U3.

9.2.5.2 Changing Device Suspend State

Device suspend is entered and exited intrinsically as part of the suspend entry and exit processes (refer to Section 10.8). The minimum device state information that shall be maintained through the duration of each Suspended USB Device State is listed in Table 9-2.

Table 9-2. Preserved USB Suspend State Parameters

Parameter	Suspended USB Device state[1, 2]	
	Address	Configured
U1_SEL/U1_PEL/U2_SEL/U2_PEL	Yes	Yes
HALT_ENDPOINT	N/A	Yes
FUNCTION REMOTE WAKEUP	Yes	Yes
Isochronous Delay	N/A	Yes
U2_Inactivity_Timeout	Yes	Yes
Force_LinkPM_Accept	Yes	Yes
U1/U2 Enable	Yes	Yes
LTM_ENABLE	Yes	Yes
Data Sequence	N/A	Yes
Hub Depth	Yes	Yes
Downstream U1_Inactivity Timeout/ U2_Inactivity Timeout (Applicable to hubs)	Yes	Yes
Port Configuration Information	Yes	Yes
Device Configuration/Interface Setting Information	N/A	Yes
Device Address	Yes	Yes
Header Sequence Number	Yes	Yes
Downstream port state	Yes	Yes

[1] No parameters other than HSN, are preserved in the *Default* Suspended USB Device State.

[2] "Yes" indicates a parameter that shall be preserved in the respective Suspended USB Device State.

Some additional Class specific device state information may also be retained during suspend.

A device shall send a Function Wake Notification after driving resume signaling (refer to Section 6.9.1 and Section 7.5.9). If the device has not been accessed for longer than tNotification (refer to Section 8.13) since sending the last Function Wake Notification, the device shall send the Function Wake Notification again until it has been accessed.

Device classes may require additional information to be retained during suspend, beyond what is identified in this specification and is beyond the scope of this specification. Devices can optionally remove power from circuitry that is not needed while in suspend.

9.2.5.3 Function Suspend

The function suspend state is a reduced power state associated with an individual function. The function may or may not be part of a composite device.

A function may be placed into function suspend independently of other functions within a composite device. A device may be transitioned into device suspend regardless of the function

suspend state of any function within the device. Function suspend state is retained while in device suspend and throughout the device suspend entry and exit processes.

9.2.5.4 Changing Function Suspend State

Functions are placed into function suspend using the FUNCTION_SUSPEND feature selector (see Table 9-7). The FUNCTION_SUSPEND feature selector also controls whether the function may initiate a function remote wakeup. Whether a function is capable of initiating a Function Remote Wake is determined by the status returned when the first interface in that function is queried using a Get Status command (refer to Section 9.4.5).

Remote wakeup (i.e., wakeup from a device suspend state) is enabled when any function within a device is enabled for function remote wakeup (note the distinction between "function remote wake" and "remote wake"). The DEVICE_REMOTE_WAKEUP feature selector is ignored and not used by SuperSpeed devices.

A function may signal that it wants to exit from function suspend by sending a Function Wake Notification to the host if it is enabled for function remote wakeup. This applies to single function devices as well as multiple function (i.e., composite) devices. If the link is in a non-U0 state, then the device must transition the link to U0 prior to sending the remote wake message. If a remote wake event occurs in multiple functions, each function shall send a Function Wake Notification. If the function has not been accessed for longer than tNotification (refer to Section 8.13) since sending the last Function Wake Notification, the function shall send the Function Wake Notification again until it has been accessed.

When all functions within a device are in function suspend and the PORT_U2_TIMEOUT field (refer to Section 10.14.2.9) is programmed to 0xFF, the device shall initiate U2 after 10 ms of link inactivity.

9.2.6 Request Processing

With the exception of SetAddress() requests (refer to Section 9.4.6), a device may begin processing a request as soon as the device receives the Setup Packet. The device is expected to "complete" processing of the request before it allows the Status stage to complete successfully. Some requests initiate operations that take many milliseconds to complete. For such requests, the device class is required to define a method other than Status stage completion to indicate that the operation has completed. For example, a reset on a hub port may take multiple milliseconds to complete depending on the status of the link attached to the port. The SetPortFeature(PORT_RESET) (refer to Section 10.14.2.9) request "completes" when the reset on the port is initiated. Completion of the reset operation is signaled when the port's status change is set to indicate that the port is now enabled. This technique prevents the host from having to poll for completion when it is known that the operation will take a relatively long period of time to complete.

9.2.6.1 Request Processing Timing

All devices are expected to handle requests in a timely manner. USB sets an upper limit of 5 seconds for any command to be processed. This limit is not applicable in all instances. The limitations are described in the following sections. It should be noted that the limitations are intended to encompass a wide range of implementations. If all devices in a USB system used the maximum allotted time for request processing, the user experience would suffer. For this reason, implementations should strive to complete requests in times that are as short as possible.

9.2.6.2 Reset/Resume Recovery Time

After a port is successfully reset or resumed, the USB system software is allowed to access the device attached to the port immediately and it is expected to respond to data transfers.

9.2.6.3 Set Address Processing

After the reset or resume, when a device receives a SetAddress() request, the device shall be able to complete processing of the request and be able to successfully complete the Status stage of the request within 50 ms. In the case of the SetAddress() request, the Status stage successfully completes when the device sends an ACK Transaction Packet in response to Status stage STATUS Transaction Packet.

After successful completion of the Status stage, the device shall be able to accept Setup packets addressed to the new address. Also, after successful completion of the Status stage, the device shall not respond to transactions sent to the old address (unless, of course, the old address and the new address are the same).

9.2.6.4 Standard Device Requests

For standard device requests that require no Data stage, a device shall be able to complete the request and be able to successfully complete the Status stage of the request within 50 ms of receipt of the request. This limitation applies to requests targeted to the device, interface, or endpoint.

For standard device requests that require a data stage transfer to the host, the device shall be able to return the first data packet to the host within 500 ms of receipt of the request. For subsequent data packets, if any, the device shall be able to return them within 500 ms of successful completion of the transmission of the previous packet. The device shall then be able to successfully complete the status stage within 50 ms after returning the last data packet.

For standard device requests that require a data stage transfer to the device, the 5-second limit applies. This means that the device shall be capable of accepting all data packets from the host and successfully completing the Status stage if the host provides the data at the maximum rate at which the device can accept it. Delays between packets introduced by the host add to the time allowed for the device to complete the request.

9.2.6.5 Class-specific Requests

Unless specifically exempted in the class document, all class-specific requests shall meet the timing limitations for standard device requests. If a class document provides an exemption, the exemption may only be specified on a request-by-request basis.

A class document may require that a device respond more quickly than is specified in this section. Faster response may be required for standard and class-specific requests.

9.2.6.6 Speed Dependent Descriptors

A device capable of operation at SuperSpeed shall be capable of operating at one of the USB 2.0 defined speeds. The device always knows its operational speed as part of connection processing (refer to Section 10.5 for more details on the connection process). A device operates at a single speed after completing the reset sequence. In particular, there is no speed switch during normal operation. However, a SuperSpeed capable device may have configurations that are speed

dependent. That is, it may have some configurations that are only possible when operating at SuperSpeed or some that are only possible when operating at high speed. SuperSpeed capable devices shall support reporting the speeds they can operate at. Note that a USB 3.0 hub is the only device that is allowed to operate at both USB 2.0 and SuperSpeed simultaneously.

A SuperSpeed capable device responds with descriptor information that is valid for the current operating speed. For example, when a device is asked for configuration descriptors, it only returns those for the current operating speed (e.g., high speed). When operating in SuperSpeed mode, the device shall report the other speeds it can operate via its BOS descriptor (refer to Section 9.6.2).

Note that when operating at USB 2.0 speeds, the device shall report the other USB 2.0 speeds it supports using the standard mechanism defined in the USB 2.0 specification in addition to reporting the other speeds supported by the device in its BOS descriptor. Devices with a value of at least 0210H in the *bcdUSB* field of their device descriptor shall support GetDescriptor (BOS Descriptor) requests.

NOTE

These descriptors are not retrieved unless the host explicitly issues the corresponding GetDescriptor requests.

9.2.7 Request Error

When a request not defined for the device is inappropriate for the current setting of the device or has values that are not compatible with the request is received, a Request Error exists. The device deals with the Request Error by returning a STALL Transaction Packet in response to the next Data stage transaction or in the Status stage of the message. It is preferred that the STALL Transaction Packet be returned at the next Data stage transaction to avoid unnecessary bus activity.

9.3 USB Device Requests

All devices respond to requests from the host on the device's Default Control Pipe. These requests are made using control transfers. The request and the request's parameters are sent to the device in the Setup packet. The host is responsible for establishing the values passed in the fields listed in Table 9-3. Every Setup packet has 8 bytes.

Table 9-3. Format of Setup Data

Offset	Field	Size	Value	Description
0	*bmRequestType*	1	Bitmap	Characteristics of request: D7: Data transfer direction 0 = Host-to-device 1 = Device-to-host D6...5: Type 0 = Standard 1 = Class 2 = Vendor 3 = Reserved D4...0: Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4...31 = Reserved
1	*bRequest*	1	Value	Specific request (refer to Table 9-4)
2	*wValue*	2	Value	Word-sized field that varies according to request
4	*wIndex*	2	Index or Offset	Word-sized field that varies according to request; typically used to pass an index or offset
6	*wLength*	2	Count	Number of bytes to transfer if there is a Data stage

9.3.1 bmRequestType

This bitmapped field identifies the characteristics of the specific request. In particular, this field identifies the direction of data transfer in the second phase of the control transfer. The state of the *Direction* bit is ignored if the *wLength* field is zero, signifying there is no Data stage.

USB 3.0 defines a series of standard requests that all devices shall support. These are listed in Table 9-4. In addition, a device class may define additional requests. A device vendor may also define requests supported by the device.

Requests may be directed to the device, an interface on the device, or a specific endpoint on a device. This field also specifies the intended recipient of the request. When an interface is specified, the *wIndex* field identifies the interface. When an endpoint is specified, the *wIndex* field identifies the endpoint.

9.3.2 bRequest

This field specifies the particular request. The *Type* bits in the *bmRequestType* field modify the meaning of this field. This specification defines values for the *bRequest* field only when the bits are reset to zero, indicating a standard request (refer to Table 9-4).

9.3.3 wValue

The contents of this field vary according to the request. It is used to pass a parameter to the device, specific to the request.

9.3.4 wIndex

The contents of this field vary according to the request. It is used to pass a parameter to the device, specific to the request.

The *wIndex* field is often used in requests to specify an endpoint or an interface. Figure 9-2 shows the format of *wIndex* when it is used to specify an endpoint.

D7	D6	D5	D4	D3	D2	D1	D0
Direction	Reserved (Reset to Zero)			Endpoint Number			
D15	D14	D13	D12	D11	D10	D7	D8
Reserved (Reset to Zero)							

U-081

Figure 9-2. wIndex Format when Specifying an Endpoint

The *Direction* bit is set to zero to indicate the OUT endpoint with the specified *Endpoint Number* and to one to indicate the IN endpoint. In the case of a control pipe, the request should have the *Direction* bit set to zero but the device may accept either value of the *Direction* bit.

Figure 9-3 shows the format of *wIndex* when it is used to specify an interface.

D7	D6	D5	D4	D3	D2	D1	D0
Interface Number							
D15	D14	D13	D12	D11	D10	D7	D8
Reserved (Reset to Zero)							

U-082

Figure 9-3. wIndex Format when Specifying an Interface

9.3.5 wLength

This field specifies the length of the data transferred during the second phase of the control transfer. The direction of data transfer (host-to-device or device-to-host) is indicated by the *Direction* bit of the *bmRequestType* field. If this field is zero, there is no data transfer phase.

On an input request, a device shall never return more data than is indicated by the *wLength* value; it may return less. On an output request, *wLength* will always indicate the exact amount of data to be sent by the host. Device behavior is undefined if the host should send more or less data than is specified in *wLength.*

9.4 Standard Device Requests

This section describes the standard device requests defined for all devices. Table 9-4 outlines the standard device requests, while Table 9-5 and Table 9-6 give the standard request codes and descriptor types, respectively.

Devices shall respond to standard device requests, even if the device has not yet been assigned an address or has not been configured. If a standard request defines a persistent parameter that can be modified, the reset/default value for that parameter unless otherwise specified is zero.

Table 9-4. Standard Device Requests

bmRequestType	bRequest	wValue	wIndex		wLength	Data
00000000B 00000001B 00000010B	CLEAR_FEATURE	Feature Selector	Zero Interface Endpoint		Zero	None
10000000B	GET_CONFIGURATION	Zero	Zero		One	Configuration Value
10000000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID		Descriptor Length	Descriptor
10000001B	GET_INTERFACE	Zero	Interface		One	Alternate Interface
10000000B 10000001B 10000010B	GET_STATUS	Zero	Zero Interface Endpoint		Two	Device, Interface, or Endpoint Status
00000000B	SET_ADDRESS	Device Address	Zero		Zero	None
00000000B	SET_CONFIGURATION	Configuration Value	Zero		Zero	None
00000000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID		Descriptor Length	Descriptor
00000000B 00000001B 00000010B	SET_FEATURE	Feature Selector	Suspend Options	Zero Interface Point	Zero	None

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000001B	SET_INTERFACE	Alternate Setting	Interface	Zero	None
00000000B	SET_ISOCH_DELAY	Delay in ns	Zero	Zero	None
00000000B	SET_SEL	Zero	Zero	Six	Exit Latency Values
10000010B	SYNCH_FRAME	Zero	Endpoint	Two	Frame Number

Table 9-5. Standard Request Codes

bRequest	Value
GET_STATUS	0
CLEAR_FEATURE	1
Reserved for future use	2
SET_FEATURE	3
Reserved for future use	4
SET_ADDRESS	5
GET_DESCRIPTOR	6
SET_DESCRIPTOR	7
GET_CONFIGURATION	8
SET_CONFIGURATION	9
GET_INTERFACE	10
SET_INTERFACE	11
SYNCH_FRAME	12
SET_SEL	48
SET_ISOCH_DELAY	49

Table 9-6. Descriptor Types

Descriptor Types	Value
DEVICE	1
CONFIGURATION	2
STRING	3
INTERFACE	4
ENDPOINT	5
Reserved	6
Reserved	7
INTERFACE_POWER[1]	8
OTG	9
DEBUG	10
INTERFACE_ASSOCIATION	11
BOS	15
DEVICE CAPABILITY	16
SUPERSPEED_USB_ENDPOINT_COMPANION	48

Feature selectors are used when enabling or setting features, such as function remote wakeup, specific to a device, interface, or endpoint. The values for the feature selectors are given in Table 9-7.

Table 9-7. Standard Feature Selectors

Feature Selector	Recipient	Value
ENDPOINT_HALT	Endpoint	0
FUNCTION_SUSPEND	Interface	0
U1_ENABLE	Device	48
U2_ENABLE	Device	49
LTM_ENABLE	Device	50
B3_NTF_HOST_REL[2]	Device	51

If an unsupported or invalid request is made to a device, the device responds by returning a STALL Transaction Packet in the Data or Status stage of the request. If the device detects the error in the Setup stage, it is preferred that the device returns a STALL Transaction Packet at the earlier of the Data or Status stage. Receipt of an unsupported or invalid request does not cause the *Halt* feature on the control pipe to be set. If, for any reason, the device becomes unable to communicate via its Default Control Pipe due to an error condition, the device shall be reset to clear the condition and restart the Default Control Pipe.

[1] The INTERFACE_POWER descriptor is defined in the current revision of the *USB Interface Power Management Specification*.
[2] This Feature Selector value shall be reserved for OTG use. Refer to Section 6.4 of the *USB 3.0 OTG and EH Supplement* for its definition.

9.4.1 Clear Feature

This request is used to clear or disable a specific feature.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B 00000001B 00000010B	CLEAR_FEATURE	Feature Selector	Zero Interface Endpoint	Zero	None

Feature selector values in *wValue* shall be appropriate to the recipient. Only device feature selector values may be used when the recipient is a device, only interface feature selector values may be used when the recipient is an interface, and only endpoint feature selector values may be used when the recipient is an endpoint.

Refer to Table 9-7 for a definition of which feature selector values are defined for which recipients.

A ClearFeature() request that references a feature that cannot be cleared, that does not exist, or that references an interface or an endpoint that does not exist, will cause the device to respond with a Request Error.

If *wLength* is non-zero, then the device behavior is not specified.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: This request is valid when the device is in the Address state; references to interfaces or to endpoints other than the Default Control Pipe shall cause the device to respond with a Request Error.

Configured state: This request is valid when the device is in the Configured state.

NOTE

The device shall process a Clear Feature (U1_Enable or U2_Enable or LTM_Enable) only if the device is in the configured state.

9.4.2 Get Configuration

This request returns the current device configuration value.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000000B	GET_CONFIGURATION	Zero	Zero	One	Configuration Value

If the returned value is zero, the device is not configured.

If *wValue*, *wIndex*, or *wLength* are not as specified above, then the device behavior is not specified.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: The value zero shall be returned.

Configured state: The non-zero *bConfigurationValue* of the current configuration shall be returned.

9.4.3 Get Descriptor

This request returns the specified descriptor if the descriptor exists.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID (refer to Section 9.6.7)	Descriptor Length	Descriptor

The *wValue* field specifies the descriptor type in the high byte (refer to Table 9-6) and the descriptor index in the low byte. The descriptor index is used to select a specific descriptor (only for configuration and string descriptors) when several descriptors of the same type are implemented in a device. For example, a device can implement several configuration descriptors. For other standard descriptors that can be retrieved via a GetDescriptor() request, a descriptor index of zero shall be used. The range of values used for a descriptor index is from 0 to one less than the number of descriptors of that type (excluding string descriptors) implemented by the device.

The *wIndex* field specifies the Language ID for string descriptors or is reset to zero for other descriptors. The *wLength* field specifies the number of bytes to return. If the descriptor is longer than the *wLength* field, only the initial bytes of the descriptor are returned. If the descriptor is shorter than the *wLength* field, the device indicates the end of the control transfer by sending a short packet when further data is requested.

The standard request to a device supports four types of descriptors: device, configuration, BOS (Binary device Object Store), and string. As noted in Section 9.2.6.6, a device operating in SuperSpeed mode reports the other speeds it supports via the BOS descriptor and shall not support the device_qualifier and other_speed_configuration descriptors. A request for a configuration descriptor returns the configuration descriptor, all interface descriptors, endpoint descriptors and endpoint companion descriptors (when operating in SuperSpeed mode) for all of the interfaces in a single request. The first interface descriptor follows the configuration descriptor. The endpoint

descriptors for the first interface follow the first interface descriptor. In addition, SuperSpeed devices shall return Endpoint Companion descriptors for each of the endpoints in that interface to return the endpoint capabilities required for SuperSpeed capable devices, which would not fit inside the existing endpoint descriptor footprint. If there are additional interfaces, their interface descriptor, endpoint descriptors, and endpoint companion descriptors (when operating in SuperSpeed mode) follow the first interface's endpoint and endpoint companion (when operating in SuperSpeed mode) descriptors.

This specification also defines a flexible and extensible framework for describing and adding device-level capabilities to the set of USB standard specifications. The BOS descriptor (refer to Section 9.6.2) defines a root descriptor that is similar to the configuration descriptor, and is the base descriptor for accessing a family of related descriptors. A host can read a BOS descriptor and learn from the *wTotalLength* field the entire size of the device-level descriptor set, or it can read in the entire BOS descriptor set of device capabilities. There is no way for a host to read individual device capability descriptors. The entire set can only be accessed via reading the BOS descriptor with a GetDescriptor() request and using the length reported in the *wTotalLength* field.

Class-specific and/or vendor-specific descriptors follow the standard descriptors they extend or modify.

All devices shall provide a device descriptor and at least one configuration descriptor. If a device does not support a requested descriptor, it responds with a Request Error.

Default state: This is a valid request when the device is in the Default state.

Address state: This is a valid request when the device is in the Address state.

Configured state: This is a valid request when the device is in the Configured state.

9.4.4 Get Interface

This request returns the selected alternate setting for the specified interface.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000001B	GET_INTERFACE	Zero	Interface	One	Alternate Setting

Some devices have configurations with interfaces that have mutually exclusive settings. This request allows the host to determine the currently selected alternate setting.

If *wValue* or *wLength* are not as specified above, then the device behavior is not specified.

If the interface specified does not exist, then the device responds with a Request Error.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: A Request Error response is given by the device.

Configured state: This is a valid request when the device is in the Configured state.

9.4.5 Get Status

This request returns status for the specified recipient.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000000B 10000001B 10000010B	GET_STATUS	Zero	Zero Interface Endpoint	Two	Device, Interface, or Endpoint Status

The *Recipient* bits of the bmRequestType field specify the desired recipient. The data returned is the current status of the specified recipient. If the recipient is an endpoint, then the lower byte of *wIndex* identifies the endpoint whose status is being queried.

If *wValue* or *wLength* are not as specified above or if *wIndex* is non-zero for a device status request, then the behavior of the device is not specified.

If an interface or an endpoint is specified that does not exist, then the device responds with a Request Error.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: If an interface or an endpoint other than the Default Control Pipe is specified, then the device responds with a Request Error.

Configured state: If an interface or an endpoint that does not exist is specified, then the device responds with a Request Error.

A GetStatus() request to a device returns the information shown in Figure 9-4.

D7	D6	D5	D4	D3	D2	D1	D0
Reserved (Reset to Zero)			LTM Enable	U2 Enable	U1 Enable	Remote Wakeup	Self-Powered
D15	D14	D13	D12	D11	D10	D7	D8
Reserved (Reset to Zero)							

U-083

Figure 9-4. Information Returned by a GetStatus() Request to a Device

The *Self Powered* field indicates whether the device is currently self-powered. If D0 is reset to zero, the device is bus-powered. If D0 is set to one, the device is self-powered. The *Self Powered* field may not be changed by the SetFeature() or ClearFeature() requests.

The *Remote Wakeup* field is reserved and must be set to zero by SuperSpeed devices. SuperSpeed devices use the *Function Remote Wake* enable/disable field to indicate whether they are enabled for Remote Wake.

The *U1 Enable* field indicates whether the device is currently enabled to initiate U1 entry. If D2 is set to zero, the device is disabled from initiating U1 entry, otherwise, it is enabled to initiate U1 entry. The *U1 Enable* field can be modified by the SetFeature() and ClearFeature() requests using the U1_ENABLE feature selector. This field is reset to zero when the device is reset.

The *U2 Enable* field indicates whether the device is currently enabled to initiate U2 entry. If D3 is set to zero, the device is disabled from initiating U2 entry otherwise it is enabled to initiate U2 entry. The *U2 Enable* field can be modified by the SetFeature() and ClearFeature() requests using the U2_ENABLE feature selector. This field is reset to zero when the device is reset.

The *LTM Enable* field indicates whether the device is currently enabled to send Latency Tolerance Messages. If D4 is set to zero, the device is disabled from sending Latency Tolerance Messages otherwise it is enabled to send Latency Tolerance Messages. The *LTM Enable* field can be modified by the SetFeature() and ClearFeature() requests using the LTM_ENABLE feature selector. This field is reset to zero when the device is reset.

A GetStatus() request to the first interface in a function returns the information shown in Figure 9-5.

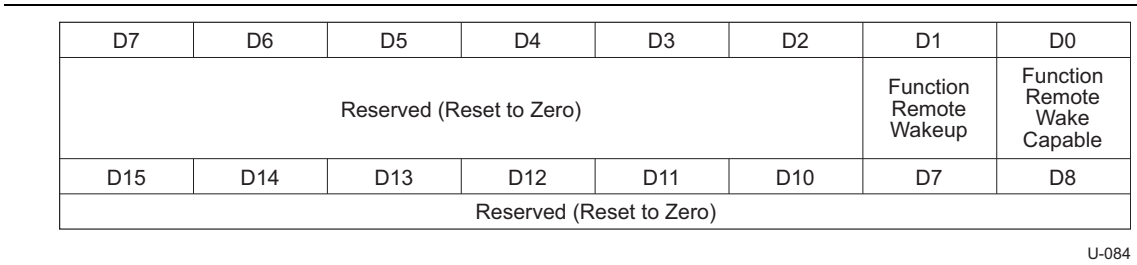


Figure 9-5. Information Returned by a GetStatus() Request to an Interface

The *Function Remote Wake Capable* field indicates whether the function supports remote wake up. The *Function Remote Wakeup* field indicates whether the function is currently enabled to request remote wakeup. The default mode for functions that support function remote wakeup is disabled. If D1 is reset to zero, the ability of the function to signal remote wakeup is disabled. If D1 is set to one, the ability of the function to signal remote wakeup is enabled. The *Function Remote Wakeup* field can be modified by the SetFeature() requests using the *FUNCTION_SUSPEND* feature selector. This *Function Remote Wakeup* field is reset to zero when the function is reset.

A GetStatus() request to any other interface in a function shall return all zeros.

A GetStatus() request to an endpoint returns the information shown in Figure 9-6.

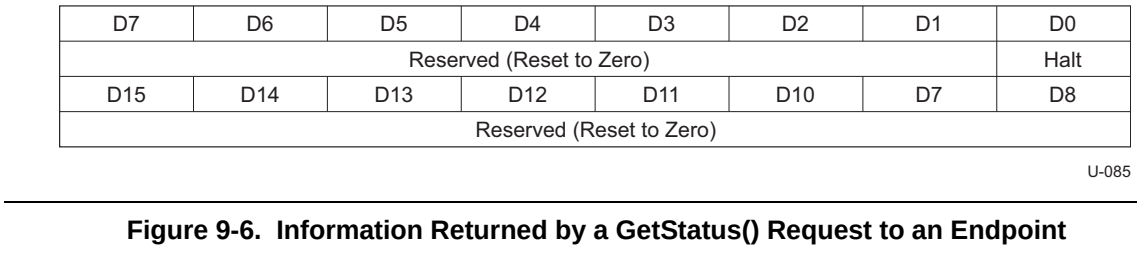


Figure 9-6. Information Returned by a GetStatus() Request to an Endpoint

The *Halt* feature is required to be implemented for all interrupt and bulk endpoint types. If the endpoint is currently halted, then the Halt feature is set to one. Otherwise, the Halt feature is reset to zero. The Halt feature may optionally be set with the SetFeature(ENDPOINT_HALT) request. When set by the SetFeature() request, the endpoint exhibits the same stall behavior as if the field

had been set by a hardware condition. If the condition causing a halt has been removed, clearing the *Halt* feature via a ClearFeature(ENDPOINT_HALT) request results in the endpoint no longer returning a STALL Transaction Packet. Regardless of whether an endpoint has the *Halt* feature set, a ClearFeature(ENDPOINT_HALT) request always results in the data sequence being reinitialized to zero, and if Streams are enabled, the Stream State Machine shall be reinitialized to the *Disabled* state. The *Halt* feature is reset to zero after either a SetConfiguration() or SetInterface() request even if the requested configuration or interface is the same as the current configuration or interface.

SuperSpeed devices do not support functional stall on control endpoints and hence do not require the *Halt* feature be implemented for any control endpoints.

9.4.6 Set Address

This request sets the device address for all future device accesses.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B	SET_ADDRESS	Device Address	Zero	Zero	None

The *wValue* field specifies the device address to use for all subsequent accesses.

The Status stage after the initial Setup packet assumes the same device address as the Setup packet. The device does not change its device address until after the Status stage of this request is completed successfully. Note that this is a difference between this request and all other requests. For all other requests, the operation indicated shall be completed before the Status stage.

If the specified device address is greater than 127, or if *wIndex* or *wLength* is non-zero, then the behavior of the device is not specified.

Default state: If the address specified is non-zero, then the device shall enter the Address state; otherwise, the device remains in the Default state (this is not an error condition).

Address state: If the address specified is zero, then the device shall enter the Default state; otherwise, the device remains in the Address state but uses the newly-specified address.

Configured state: Device behavior when this request is received while the device is in the Configured state is not specified.

9.4.7 Set Configuration

This request sets the device configuration.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B	SET_CONFIGURATION	Configuration Value	Zero	Zero	None

The lower byte of the *wValue* field specifies the desired configuration. This configuration value shall be zero or match a configuration value from a configuration descriptor. If the configuration value is zero, the device is placed in its Address state. The upper byte of the *wValue* field is reserved.

If *wIndex*, *wLength*, or the upper byte of *wValue* is non-zero, then the behavior of this request is not specified.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: If the specified configuration value is zero, then the device remains in the Address state. If the specified configuration value matches the configuration value from a configuration descriptor, then that configuration is selected and the device enters the Configured state. Otherwise, the device responds with a Request Error.

Configured state: If the specified configuration value is zero, then the device enters the Address state. If the specified configuration value matches the configuration value from a configuration descriptor, then that configuration is selected and the device remains in the Configured state. Otherwise, the device responds with a Request Error.

9.4.8 Set Descriptor

This request is optional and may be used to update existing descriptors or new descriptors may be added.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Language ID (refer to Section 9.6.7) or zero	Descriptor Length	Descriptor

The *wValue* field specifies the descriptor type in the high byte (refer to Table 9-6) and the descriptor index in the low byte. The descriptor index is used to select a specific descriptor (only for configuration and string descriptors) when several descriptors of the same type are implemented in a device. For example, a device can implement several configuration descriptors. For other standard descriptors that can be set via a SetDescriptor() request, a descriptor index of zero shall be used. The range of values used for a descriptor index is from 0 to one less than the number of descriptors of that type (excluding string descriptors) implemented by the device.

The *wIndex* field specifies the Language ID for string descriptors or is reset to zero for other descriptors. The wLength field specifies the number of bytes to transfer from the host to the device.

The only allowed values for descriptor type are device, configuration, and string descriptor types.

If this request is not supported, the device will respond with a Request Error.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: If supported, this is a valid request when the device is in the Address state.

Configured state: If supported, this is a valid request when the device is in the Configured state.

9.4.9 Set Feature

This request is used to set or enable a specific feature.

bmRequestType	bRequest	wValue	wIndex		wLength	Data
00000000B 00000001B 00000010B	SET_FEATURE	Feature Selector	Suspend Options	Zero Interface Point	Zero	None

Feature selector values in *wValue* shall be appropriate to the recipient. Only device feature selector values may be used when the recipient is a device; only interface feature selector values may be used when the recipient is an interface; and only endpoint feature selector values may be used when the recipient is an endpoint. If the recipient is an endpoint, then the lower byte of *wIndex* identifies the endpoint.

Refer to Table 9-7 for a definition of which feature selector values are defined for which recipients.

The FUNCTION_SUSPEND feature is only defined for an interface recipient. The lower byte of *wIndex* shall be set to the first interface that is part of that function.

The U1/U2_ENABLE feature is only defined for a device recipient and wIndex shall be set to zero. Setting the U1/U2_ENABLE feature allows the device to initiate U1/U2 entry respectively. A device shall support the U1/U2ENABLE feature when in the Configured SuperSpeed state only. System software must not enable the device to initiate U1 if the time for U1 System Exit Latency initiated by Host plus one Bus Interval time is greater than the minimum of the service intervals of any periodic endpoints in the device. In addition, system software must not enable the device to initiate U2 if the time for U2 System Exit Latency initiated by Host plus one Bus Interval time is greater than the minimum of the service intervals of any periodic endpoints in the device.

The LTM_ENABLE feature is only defined for a device recipient and *wIndex* shall be set to zero. Setting the LTM_ENABLE feature allows the device to send Latency Tolerance Messages. A device shall support the LTM_ENABLE feature if it is in the Configured SuperSpeed state and supports the LTM capability.

A SetFeature() request that references a feature that cannot be set or that does not exist causes a STALL Transaction Packet to be returned in the Status stage of the request.

Table 9-8. Suspend Options

Bit	Description
0	**Value** **Meaning** 0 Normal operation state (default) 1 Low power suspend state
1	**Value** **Meaning** 0 Function Remote Wake Disabled (Default) 1 Function Remote Wake Enabled
2-7	Reserved

If the feature selector is *FUNCTION_SUSPEND*, then the most significant byte of *wIndex* is used to specify Suspend options. The recipient of a SetFeature (FUNCTION_SUSPEND…) shall be the first interface in the function; and, hence, the *bmRequestType* shall be set to one. The valid encodings for the *FUNCTION_SUSPEND* suspend options are listed in Table 9-8.

If *wLength* is non-zero, then the behavior of the device is not specified.

If an endpoint or interface is specified that does not exist, then the device responds with a Request Error.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: If an interface or an endpoint other than the Default Control Pipe is specified then the device responds with a Request Error. If the device receives a SetFeature(U1/U2 Enable or LTM Enable or FUNCTION_SUSPEND), then the device responds with a Request Error.

Configured state: This is a valid request when the device is in the Configured state.

9.4.10 Set Interface

This request allows the host to select an alternate setting for the specified interface.

bmRequestType	bRequest	WValue	wIndex	wLength	Data
00000001B	SET_INTERFACE	Alternate Setting	Interface	Zero	None

Some devices have configurations with interfaces that have mutually exclusive settings. This request allows the host to select the desired alternate setting. If a device only supports a default setting for the specified interface, then a STALL Transaction Packet may be returned in the Status stage of the request. This request cannot be used to change the set of configured interfaces (the SetConfiguration() request shall be used instead).

If the interface or the alternate setting does not exist, then the device responds with a Request Error. If *wLength* is non-zero, then the behavior of the device is not specified.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: The device shall respond with a Request Error.

Configured state: This is a valid request when the device is in the Configured state.

9.4.11 Set Isochronous Delay

This request informs the device of the delay from the time a host transmits a packet to the time it is received by the device.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B	SET_ISOCH_DELAY	Delay in ns	Zero	Zero	None

The *wValue* field specifies a delay from 0 to 65535 ns. This delay represents the time from when the host starts transmitting the first framing symbol of the packet to when the device receives the first framing symbol of that packet.

If *wIndex* or *wLength* is non-zero, then the behavior of this request is not specified.

Default state: This is a valid request when the device is in the Default state.

Address state: This is a valid request when the device is in the Address state.

Configured state: This is a valid request when the device is in the Configured state.

9.4.12 Set SEL

This request sets both the U1 and U2 System Exit Latency and the U1 or U2 exit latency for all the links between a device and a root port on the host.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B	SET_SEL	Zero	Zero	Six	Exit Latency Values

The latency values are sent to the device in the data stage of the control transfer in the following format:

Offset	Name	Meaning
0	U1SEL	Time in μs for U1 System Exit Latency
1	U1PEL	Time in μs for U1 Device to Host Exit Latency
2	U2SEL	Time in μs for U2 System Exit Latency
4	U2PEL	Time in μs for U2 Device to Host Exit Latency

Figure C-2 in Appendix C illustrates the total latency a device may experience. The components of latency include the following:

- t1: the time to transition all links in the path to the host to U0 when the transition is initiated by the device
- t2: the time for the ERDY to traverse the interconnect hierarchy from the device to the host
- t3: the time for the host to consume the ERDY and transmit a response to that request
- t4: the time for the response to traverse the interconnect hierarchy from the host to the device

The U1SEL and U2SEL values represent the total round trip path latency when transitioning the links between the device and host from U1 or U2 respectively to U0 under worst case circumstances when the transition is initiated by the device. This is the sum of times t1, t2, and t4.

The U1PEL and U2PEL values represent the device to host latency to transition the entire path of links between the device and host from U1 or U2 respectively to U0 under worst case circumstances when the transition is initiated by the device. This time includes only t1.

For more information, refer to Section C.1.5.1.

If *wIndex* or *wValue* is not set to zero or *wLength* is not six, then the behavior of the device is not specified.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: This is a valid request when the device is in the Address state.

Configured state: This is a valid request when the device is in the Configured state.

9.4.13 Synch Frame

This request is used to set and then report an endpoint's synchronization frame.

bmRequestType	bRequest	WValue	wIndex	wLength	Data
10000010B	synch_frame	Zero	Endpoint	Two	Frame Number

When an endpoint supports isochronous transfers, the endpoint may also require per-frame transfers to vary in size according to a specific pattern. The host and the endpoint must agree on which frame the repeating pattern begins. The number of the frame in which the pattern began is returned to the host.

If a SuperSpeed device supports the Synch Frame request, it shall internally synchronize itself to the zeroth microframe and have a time notion of classic frame. Only the frame number is used to synchronize and reported by the device endpoint (i.e., no microframe number). The endpoint must synchronize to the zeroth microframe.

This value is only used for isochronous data transfers using implicit pattern synchronization. If *wValue* is non-zero or *wLength* is not two, then the behavior of the device is not specified.

If the specified endpoint does not support this request, then the device will respond with a Request Error.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: The device shall respond with a Request Error.

Configured state: This is a valid request when the device is in the Configured state.

9.4.14 Events and Their Effect on Device Parameters

This section lists the various parameters and the effect on those parameters when the device receives a control transfer command or when it observes a bus reset on the bus. An X denotes that the parameter is reset to its default value when the said event occurs. A Y denotes that the particular Parameter is modified by the event.

Control transfers and events not identified in the table shall not affect the value of parameters shown in Table 9-9.

Table 9-9. Device Parameters and Events

Parameter	Event							
	Warm Reset	Hot Reset	Set Address 0	Set Address	Set Configuration	Set Interface	ClearFeature (STALL)	Disconnect
Device Address	X	X	X	Y				X
Device Configuration Value	X	X	X		Y			X
Alternate Interface Setting	X	X	X		X	Y		X

Parameter	Event							
	Warm Reset	**Hot Reset**	**Set Address 0**	**Set Address**	**Set Configuration**	**Set Interface**	**ClearFeature (STALL)**	**Disconnect**
U1_SEL/U1_PEL/ U2_SEL/U2_PEL	X	X	X					X
HALT_ENDPOINT	X	X	X		X	X (if the EP is affected)	X	X
FUNCTION REMOTE WAKEUP	X	X	X		X	X		X
Isochronous Delay	X	X	X					X
U2_Inactivity_Timeout	X	X	X					X
Force_LinkPM_Accept	X	X	X					X
U1/U2 Enable	X	X	X					X
LTM_ENABLE	X	X	X					X
HeaderSequence Number related to DPs	X	X	X		X	X (if the EP is affected)	X	X
Hub Depth	X	X	X		X			X
Downstream U1_Inactivity Timeout/ U2_Inactivity Timeout (Applicable to host and hub)	X	X	X					X (Hub Upstream or Hub/Host Downstream)
Port Configuration Information	X							X

9.5 Descriptors

Devices report their attributes using descriptors. A descriptor is a data structure with a defined format. Each descriptor begins with a byte-wide field that contains the total number of bytes in the descriptor followed by a byte-wide field that identifies the descriptor type.

Using descriptors allows concise storage of the attributes of individual configurations because each configuration may reuse descriptors or portions of descriptors from other configurations that have the same characteristics. In this manner, the descriptors resemble individual data records in a relational database.

Where appropriate, descriptors contain references to string descriptors that provide displayable information describing a descriptor in human-readable form. The inclusion of string descriptors is optional. However, the reference fields within descriptors are mandatory. If a device does not support string descriptors, string reference fields shall be reset to zero to indicate no string descriptor is available.

If a descriptor returns with a value in its length field that is less than defined by this specification, the descriptor is invalid and should be rejected by the host. If the descriptor returns with a value in its length field that is greater than defined by this specification, the extra bytes are ignored by the host, but the next descriptor is located using the length returned rather than the length expected.

A device may return class- or vendor-specific descriptors in two ways:

1. If the class or vendor specific descriptors use the same format as standard descriptors (e.g., start with a length byte and followed by a type byte), they shall be returned interleaved with standard descriptors in the configuration information returned by a GetDescriptor(Configuration) request. In this case, the class or vendor-specific descriptors shall follow a related standard descriptor they modify or extend.
2. If the class or vendor specific descriptors are independent of configuration information or use a non-standard format, a GetDescriptor() request specifying the class or vendor specific descriptor type and index may be used to retrieve the descriptor from the device. A class or vendor specification will define the appropriate way to retrieve these descriptors.

9.6 Standard USB Descriptor Definitions

The standard descriptors defined in this specification may only be modified or extended by revision of this specification.

9.6.1 Device

A device descriptor describes general information about a device. It includes information that applies globally to the device and all of the device's configurations. A device has only one device descriptor.

The device descriptor of a SuperSpeed capable device operating in SuperSpeed mode has a version number of 3.0 (0300H). The device descriptor of a SuperSpeed capable device operating in one of the USB 2.0 modes has a version number of 2.1 (0210H).

The *bcdUSB* field contains a BCD version number. The value of the *bcdUSB* field is 0xJJMN for version JJ.M.N (JJ – major version number, M – minor version number, N – sub-minor version number), e.g., version 2.1.3 is represented with value 0x0213 and version 3.0 is represented with a value of 0x0300.

The *bNumConfigurations* field indicates the number of configurations at the current operating speed. Configurations for the other operating speed are not included in the count. If there are specific configurations of the device for specific speeds, the *bNumConfigurations* field only reflects the number of configurations for a single speed, not the total number of configurations for both speeds.

If the device is operating at SuperSpeed, the *bMaxPacketSize0* field shall be set to 09H (see Table 9-10) indicating a 512-byte maximum packet. SuperSpeed operation does not allow other maximum packet sizes for the default control pipe (endpoint 0) control endpoint.

All devices have a default control pipe. The maximum packet size of a device's default control pipe is described in the device descriptor. Endpoints specific to a configuration and its interface(s) are described in the configuration descriptor. A configuration and its interface(s) do not include an endpoint descriptor for the default control pipe. Other than the maximum packet size, the characteristics of the default control pipe are defined by this specification and are the same for all SuperSpeed devices.

The *bNumConfigurations* field identifies the number of configurations the device supports. Table 9-10 shows the standard device descriptor.

Table 9-10. Standard Device Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	DEVICE Descriptor Type
2	*bcdUSB*	2	BCD	USB Specification Release Number in Binary-Coded Decimal (i.e., 2.10 is 210H). This field identifies the release of the USB Specification with which the device and its descriptors are compliant.
4	*bDeviceClass*	1	Class	Class code (assigned by the USB-IF). If this field is reset to zero, each interface within a configuration specifies its own class information and the various interfaces operate independently. If this field is set to a value between 1 and FEH, the device supports different class specifications on different interfaces and the interfaces may not operate independently. This value identifies the class definition used for the aggregate interfaces. If this field is set to FFH, the device class is vendor-specific.
5	*bDeviceSubClass*	1	SubClass	Subclass code (assigned by the USB-IF). These codes are qualified by the value of the *bDeviceClass* field. If the *bDeviceClass* field is reset to zero, this field shall also be reset to zero. If the *bDeviceClass* field is not set to FFH, all values are reserved for assignment by the USB-IF.
6	*bDeviceProtocol*	1	Protocol	Protocol code (assigned by the USB-IF). These codes are qualified by the value of the *bDeviceClass* and the *bDeviceSubClass* fields. If a device supports class-specific protocols on a device basis as opposed to an interface basis, this code identifies the protocols that the device uses as defined by the specification of the device class. If this field is reset to zero, the device does not use class-specific protocols on a device basis. However, it may use class-specific protocols on an interface basis. If this field is set to FFH, the device uses a vendor-specific protocol on a device basis.

Offset	Field	Size	Value	Description
7	*bMaxPacketSize0*	1	Number	Maximum packet size for endpoint zero. The bMaxPacketSize0 value is used as the exponent for a $2^{bMaxPacketSize0}$ value; e.g., a bMaxPacketSize0 of 4 means a Max Packet size of 16 ($2^4 \rightarrow 16$). 09H is the only valid value in this field when operating in SuperSpeed mode.
8	*idVendor*	2	ID	Vendor ID (assigned by the USB-IF)
10	*idProduct*	2	ID	Product ID (assigned by the manufacturer)
12	*bcdDevice*	2	BCD	Device release number in binary-coded decimal
14	*iManufacturer*	1	Index	Index of string descriptor describing manufacturer
15	*iProduct*	1	Index	Index of string descriptor describing product
16	*iSerialNumber*	1	Index	Index of string descriptor describing the device's serial number
17	*bNumConfigurations*	1	Number	Number of possible configurations

9.6.2 Binary Device Object Store (BOS)

This section defines a flexible and extensible framework for describing and adding device-level capabilities to the set of USB standard specifications. As mentioned above, there exists a device descriptor, but all device-level capability extensions are defined using the following framework.

The BOS descriptor defines a root descriptor that is similar to the configuration descriptor, and is the base descriptor for accessing a family of related descriptors. A host can read a BOS descriptor and learn from the *wTotalLength* field the entire size of the device-level descriptor set, or it can read in the entire BOS descriptor set of device capabilities. The host accesses this descriptor using the GetDescriptor() request. The descriptor type in the GetDescriptor() request is set to BOS (see Table 9-11). There is no way for a host to read individual device capability descriptors. The entire set can only be accessed via reading the BOS descriptor with a GetDescriptor() request and using the length reported in the *wTotalLength* field.

Table 9-11. BOS Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	BOS Descriptor type
2	*wTotalLength*	2	Number	Length of this descriptor and all of its sub descriptors
4	*bNumDeviceCaps*	1	Number	The number of separate device capability descriptors in the BOS

Individual technology-specific or generic device-level capabilities are reported via Device Capability descriptors. The format of the Device Capability descriptor is defined in Table 9-12. The Device Capability descriptor has a generic header, with a sub-type field (*bDevCapabilityType*) which defines the layout of the remainder of the descriptor. The codes for *bDevCapabilityType* are defined in Table 9-13.

Table 9-12. Format of a Device Capability Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor.

Offset	Field	Size	Value	Description
1	*bDescriptorType*	1	Constant	Descriptor type: DEVICE CAPABILITY Type.
2	*bDevCapabilityType*	1	Number	Valid values are listed in Table 9-13.
3	*Capability-Dependent*	Var	Variable	Capability-specific format.

Device Capability descriptors are always returned as part of the BOS information returned by a GetDescriptor(BOS) request. A Device Capability cannot be directly accessed with a GetDescriptor() or SetDescriptor() request.

Table 9-13. Device Capability Type Codes

Capability Code	Value	Description
Wireless_USB	01H	Defines the set of Wireless USB-specific device level capabilities
USB 2.0 EXTENSION	02H	USB 2.0 Extension Descriptor
SUPERSPEED_USB	03H	Defines the set of SuperSpeed USB specific device level capabilities
CONTAINER_ID	04H	Defines the instance unique ID used to identify the instance across all operating modes
Reserved	00H, 05-FFH	Reserved for future use

The following section defines the USB 2.0 Extension Descriptor, the SuperSpeed USB Device Capability, and the Container ID (if supported) that a USB 3.0 device shall return when operating in SuperSpeed mode or in any of the USB 2.0 modes.

9.6.2.1 USB 2.0 Extension

A SuperSpeed device shall include the USB 2.0 Extension descriptor and shall support LPM when operating in USB 2.0 High-Speed mode.

Table 9-14. USB 2.0 Extension Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	DEVICE CAPABILITY Descriptor type
2	*bDevCapabilityType*	1	Constant	Capability type: USB 2.0 EXTENSION
3	*bmAttributes*	4	Bitmap	Bitmap encoding of supported device level features. A value of one in a bit location indicates a feature is supported; a value of zero indicates it is not supported. Encodings are: Bit — Encoding 0 — Reserved. Shall be set to zero. 1 — LPM. A value of one in this bit location indicates that this device supports the Link Power Management protocol. SuperSpeed devices shall set this bit to one. 31:2 — Reserved. Shall be set to zero.

9.6.2.2 SuperSpeed USB Device Capability

This section defines the required device-level capabilities descriptor which shall be implemented by all SuperSpeed devices. This capability descriptor cannot be directly accessed with a GetDescriptor() or SetDescriptor() request.

Table 9-15. SuperSpeed Device Capabilities Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	DEVICE CAPABILITY Descriptor type
2	*bDevCapabilityType*	1	Constant	Capability type: SUPERSPEED_USB
3	*bmAttributes*	1	Bitmap	Bitmap encoding of supported device level features. A value of one in a bit location indicates a feature is supported; a value of zero indicates it is not supported. Encodings are: Bit — Encoding 0 — Reserved. Shall be set to zero. 1 — LTM Capable. A value of one in this bit location indicates that this device has is capable of generating Latency Tolerance Messages. 7:2 — Reserved. Shall be set to zero.
4	*wSpeedsSupported*	2	Bitmap	Bitmap encoding of the speed supported by this device when operating in SuperSpeed mode. Bit — Encoding 0 — If this bit is set, then the device supports operation at low-Speed USB. 1 — If this bit is set, then the device supports operation at full-Speed USB. 2 — If this bit is set, then the device supports operation at high-Speed USB. 3 — If this bit is set, then the device supports operation at 5 Gbps. 15:4 — Reserved. Shall be set to zero.
6	*bFunctionalitySupport*	1	Number	The lowest speed at which all the functionality supported by the device is available to the user. For example if the device supports all its functionality when connected at full speed and above then it sets this value to 1. Refer to the *wSpeedsSupported* field for valid values that can be placed in this field.

Offset	Field	Size	Value	Description
7	*bU1DevExitLat*	1	Number	U1 Device Exit Latency. Worst case latency to transition from U1 to U0, assuming the latency is limited only by the device and not the device's link partner. This field applies only to the exit latency associated with an individual port, and does not apply to the total latency through a hub (e.g., from downstream port to upstream port). The following are permissible values: Value — Meaning 00H — Zero. 01H — Less than 1 µs 02H — Less than 2 µs 03H — Less than 3 µs 04H — Less than 4 µs … — … 0AH — Less than 10 µs 0BH – FFH — Reserved For a hub, this is the value for both its upstream and downstream ports.
8	*wU2DevExitLat*	2	Number	U2 Device Exit Latency. Worst case latency to transition from U2 to U0, assuming the latency is limited only by the device and not the device's link partner. Applies to all ports on a device. The following are permissible values: Value — Meaning 0000H — Zero 0001H — Less than 1 µs 0002H — Less than 2 µs 0003H — Less than 3 µs 0004H — Less than 4 µs … — … 07FFH — Less than 2047 µs 0800H – FFFFH — Reserved For a hub, this is the value for both its upstream and downstream ports.

9.6.2.3 Container ID

This section defines the device-level Container ID descriptor which shall be implemented by all USB 3.0 hubs, and is optional for other devices. If this descriptor is provided when operating in one mode, it shall be provided when operating in any mode. This descriptor may be used by a host in order to identify a unique device instance across all operating modes. If a device can also connect to a host through other technologies, the same Container ID value contained in this descriptor should also be provided over those other technologies in a technology specific manner.

This capability descriptor cannot be directly accessed with a GetDescriptor() or SetDescriptor() request.

Table 9-16. Container ID Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	DEVICE CAPABILITY Descriptor type
2	*bDevCapabilityType*	1	Constant	Capability type: CONTAINER_ID
3	*bReserved*	1	Number	This field is reserved and shall be set to zero.
4	*ContainerID*	16	UUID	This is a 128-bit number that is unique to a device instance that is used to uniquely identify the device instance across all modes of operation. This same value may be provided over other technologies as well to allow the host to identify the device independent of means of connectivity. Refer to IETF RFC 4122 for details on generation of a UUID.

9.6.3 Configuration

The configuration descriptor describes information about a specific device configuration. The descriptor contains a *bConfigurationValue* field with a value that, when used as a parameter to the SetConfiguration() request, causes the device to assume the described configuration.

The descriptor describes the number of interfaces provided by the configuration. Each interface may operate independently. For example, a Video Class device might be configured with two interfaces, each providing 64-MBps bi-directional channels that have separate data sources or sinks on the host. Another configuration might present the Video Class device as a single interface, bonding the two channels into one 128-MBps bi-directional channel.

When the host requests the configuration descriptor, all related interface, endpoint, and endpoint companion descriptors are returned (refer to Section 9.4.3).

A device has one or more configuration descriptors. Each configuration has one or more interfaces and each interface has zero or more endpoints. An endpoint is not shared among interfaces within a single configuration unless the endpoint is used by alternate settings of the same interface. Endpoints may be shared among interfaces that are part of different configurations without this restriction.

Once configured, devices may support limited adjustments to the configuration. If a particular interface has alternate settings, an alternate may be selected after configuration. Table 9-17 shows the standard configuration descriptor.

Table 9-17. Standard Configuration Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	CONFIGURATION Descriptor Type
2	*wTotalLength*	2	Number	Total length of data returned for this configuration. Includes the combined length of all descriptors (configuration, interface, endpoint, and class- or vendor-specific) returned for this configuration
4	*bNumInterfaces*	1	Number	Number of interfaces supported by this configuration
5	*bConfigurationValue*	1	Number	Value to use as an argument to the SetConfiguration() request to select this configuration
6	*iConfiguration*	1	Index	Index of string descriptor describing this configuration
7	*bmAttributes*	1	Bitmap	Configuration characteristics: D7: Reserved (set to one) D6: Self-powered D5: Remote Wakeup D4...0: Reserved (reset to zero) D7 is reserved and shall be set to one for historical reasons. A device configuration that uses power from the bus and a local source reports a non-zero value in *bMaxPower* to indicate the amount of bus power required and sets D6. The actual power source at runtime may be determined using the GetStatus(DEVICE) request (refer to Section 9.4.5). If a device configuration supports remote wakeup, D5 is set to one.
8	*bMaxPower*	1	mA	Maximum power consumption of the device from the bus in this specific configuration when the device is fully operational. Expressed in 2-mA units when the device is operating in high-speed mode and in 8-mA units when operating in SuperSpeed mode. (i.e., 50 = 100 mA in high-speed mode and 50 = 400 mA in SuperSpeed mode). Note: A device configuration reports whether the configuration is bus-powered or self-powered. Device status reports whether the device is currently self-powered. If a device is disconnected from its external power source, it updates device status to indicate that it is no longer self-powered. A device may not increase its power draw from the bus, when it loses its external power source, beyond the amount reported by its configuration. If a device can continue to operate when disconnected from its external power source, it continues to do so. If the device cannot continue to operate, it shall return to the Powered state.

9.6.4 Interface Association

The Interface Association Descriptor is used to describe that two or more interfaces are associated to the same function. An "association" includes two or more interfaces and all of their alternate setting interfaces. A device must use an Interface Association descriptor for each device function that requires more than one interface. An Interface Association descriptor is always returned as part of the configuration information returned by a GetDescriptor(Configuration) request. An interface association descriptor cannot be directly accessed with a GetDescriptor() or SetDescriptor() request. An interface association descriptor must be located before the set of interface descriptors (including all alternate settings) for the interfaces it associates. All of the interface numbers in the set of associated interfaces must be contiguous. Table 9-18 shows the standard interface association descriptor. The interface association descriptor includes function class, subclass, and protocol fields. The values in these fields can be the same as the interface class, subclass, and protocol values from any one of the associated interfaces. The preferred implementation, for existing device classes, is to use the interface class, subclass, and protocol field values from the first interface in the list of associated interfaces.

Table 9-18. Standard Interface Association Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	INTERFACE ASSOCIATION Descriptor
2	*bFirstInterface*	1	Number	Interface number of the first interface that is associated with this function
3	*bInterfaceCount*	1	Number	Number of contiguous interfaces that are associated with this function
4	*bFunctionClass*	1	Class	Class code (assigned by USB-IF). A value of zero is not allowed in this descriptor. If this field is FFH, the function class is vendor-specific. All other values are reserved for assignment by the USB-IF.
5	*bFunctionSubClass*	1	SubClass	Subclass code (assigned by USB-IF). If the *bFunctionClass* field is not set to FFH, all values are reserved for assignment by the USB-IF.
6	*bFunctionProtocol*	1	Protocol	Protocol code (assigned by USB-IF). These codes are qualified by the values of the *bFunctionClass* and *bFunctionSubClass* fields.
7	*iFunction*	1	Index	Index of string descriptor describing this function

Note: Since this particular feature was not included in earlier versions of the USB specification, there is an issue with how existing USB operating system implementations will support devices that use this descriptor. It is strongly recommended that device implementations utilizing the interface association descriptor use the Multi-interface Function Class codes in the device descriptor. This allows simple and easy identification of these devices and allows on some operating systems, installation of an upgrade driver that can parse and enumerate configurations that include the Interface Association Descriptor. The Multi-interface Function Class code is documented at *http://www.usb.org/developers/docs*.

9.6.5 Interface

The interface descriptor describes a specific interface within a configuration. A configuration provides one or more interfaces, each with zero or more endpoint descriptors. When a configuration supports more than one interface, the endpoint descriptors for a particular interface follow the interface descriptor in the data returned by the GetConfiguration() request. As mentioned earlier in this chapter, SuperSpeed devices shall return Endpoint Companion descriptors for each of the endpoints in that interface to return additional information about its endpoint capabilities. The Endpoint Companion descriptor shall immediately follow the endpoint descriptor it is associated with in the configuration information. An interface descriptor is always returned as part of a configuration descriptor. Interface descriptors cannot be directly accessed with a GetDescriptor() or SetDescriptor() request.

An interface may include alternate settings that allow the endpoints and/or their characteristics to be varied after the device has been configured. The default setting for an interface is always alternate setting zero. The SetInterface() request is used to select an alternate setting or to return to the default setting. The GetInterface() request returns the selected alternate setting.

Alternate settings allow a portion of the device configuration to be varied while other interfaces remain in operation. If a configuration has alternate settings for one or more of its interfaces, a separate interface descriptor and its associated endpoint and endpoint companion (when reporting its SuperSpeed configuration) descriptors are included for each setting.

If a device configuration supported a single interface with two alternate settings, the configuration descriptor would be followed by an interface descriptor with the *bInterfaceNumber* and *bAlternateSetting* fields set to zero and then the endpoint and endpoint companion (when reporting its SuperSpeed configuration) descriptors for that setting, followed by another interface descriptor and its associated endpoint and endpoint companion descriptors. The second interface descriptor's *bInterfaceNumber* field would also be set to zero, but the *bAlternateSetting* field of the second interface descriptor would be set to one.

If an interface uses only the Default Control Pipe, no endpoint descriptors follow the interface descriptor. In this case, the *bNumEndpoints* field shall be set to zero.

An interface descriptor never includes the Default Control Pipe in the number of endpoints. Table 9-19 shows the standard interface descriptor.

Table 9-19. Standard Interface Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	INTERFACE Descriptor Type
2	*bInterfaceNumber*	1	Number	Number of this interface. Zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
3	*bAlternateSetting*	1	Number	Value used to select this alternate setting for the interface identified in the prior field
4	*bNumEndpoints*	1	Number	Number of endpoints used by this interface (excluding the Default Control Pipe). If this value is zero, this interface only uses the Default Control Pipe.
5	*bInterfaceClass*	1	Class	Class code (assigned by the USB-IF). A value of zero is reserved for future standardization. If this field is set to FFH, the interface class is vendor-specific. All other values are reserved for assignment by the USB-IF.
6	*bInterfaceSubClass*	1	SubClass	Subclass code (assigned by the USB-IF). These codes are qualified by the value of the *bInterfaceClass* field. If the *bInterfaceClass* field is reset to zero, this field shall also be reset to zero. If the *bInterfaceClass* field is not set to FFH, all values are reserved for assignment by the USB-IF.
7	*bInterfaceProtocol*	1	Protocol	Protocol code (assigned by the USB). These codes are qualified by the value of the *bInterfaceClass* and the *bInterfaceSubClass* fields. If an interface supports class-specific requests, this code identifies the protocols that the device uses as defined by the specification of the device class. If this field is reset to zero, the device does not use a class-specific protocol on this interface. If this field is set to FFH, the device uses a vendor-specific protocol for this interface.
8	*iInterface*	1	Index	Index of string descriptor describing this interface

9.6.6 Endpoint

Each endpoint used for an interface has its own descriptor. This descriptor contains the information required by the host to determine the bandwidth requirements of each endpoint. An endpoint descriptor is always returned as part of the configuration information returned by a GetDescriptor(Configuration) request. An endpoint descriptor cannot be directly accessed with a GetDescriptor() or SetDescriptor() request. There is never an endpoint descriptor for endpoint zero. Table 9-20 shows the standard endpoint descriptor.

Table 9-20. Standard Endpoint Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	ENDPOINT Descriptor Type
2	*bEndpointAddress*	1	Endpoint	The address of the endpoint on the device described by this descriptor. The address is encoded as follows: Bit 3...0: The endpoint number Bit 6...4: Reserved, reset to zero Bit 7: Direction, ignored for control endpoints 0 = OUT endpoint 1 = IN endpoint

Offset	Field	Size	Value	Description
3	*bmAttributes*	1	Bitmap	This field describes the endpoint's attributes when it is configured using the *bConfigurationValue*. Bits 1..0: Transfer Type 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt If an interrupt endpoint, bits 5..2 are defined as follows: Bits 3..2: Reserved Bits 5..4: Usage Type 00 = Periodic 01 = Notification 10 = Reserved 11 = Reserved If isochronous, they are defined as follows: Bits 3..2: Synchronization Type 00 = No Synchronization 01 = Asynchronous 10 = Adaptive 11 = Synchronous Bits 5..4: Usage Type 00 = Data endpoint 01 = Feedback endpoint 10 = Implicit feedback Data endpoint 11 = Reserved If not an isochronous or interrupt endpoint, bits 5..2 are reserved and shall be set to zero. All other bits are reserved and shall be reset to zero. Reserved bits shall be ignored by the host.
4	*wMaxPacketSize*	2	Number	Maximum packet size this endpoint is capable of sending or receiving when this configuration is selected. For control endpoints this field shall be set to 512. For bulk endpoint types this field shall be set to 1024. For interrupt and isochronous endpoints this field shall be set to 1024 if this endpoint defines a value in the **bMaxBurst** field greater than zero. If the value in the **bMaxBurst** field is set to zero then this field can have any value from 0 to 1024 for an isochronous endpoint and 1 to 1024 for an interrupt endpoint.
6	*bInterval*	1	Number	Interval for servicing the endpoint for data transfers. Expressed in 125-µs units. For SuperSpeed isochronous and interrupt endpoints, this value shall be in the range from 1 to 16. However, the valid ranges are 8 to 16 for Notification type Interrupt endpoints. The bInterval value is used as the exponent for a $2^{(bInterval-1)}$ value; e.g., a bInterval of 4 means a period of 8 ($2^{(4-1)} \rightarrow 2^3 \rightarrow 8$). This field is reserved and shall not be used for SuperSpeed bulk or control endpoints.

The *bmAttributes* field provides information about the endpoint's Transfer Type (bits 1..0) and Synchronization Type (bits 3..2). For interrupt endpoints, the Usage Type bits (bits 5..4) indicate whether the endpoint is used for infrequent notifications that can tolerate varying latencies (bits 5..4 = 01b), or if it regularly transfers data in consecutive service intervals or is dependent on bounded latencies (bits 5..4 = 00b). For example, a hub's interrupt endpoint would specify that it is a notification type, while a mouse would specify a periodic type. For endpoints that sometimes operate in infrequent notification mode and at other times operate in periodic mode then this field shall be set to Periodic (bits 5..4 = 00b). These values may be used by software to determine appropriate power management settings. See Appendix C for details on how this value may impact power management. In addition, for isochronous endpoints the Usage Type bit (bits 5..4) indicate whether this is an endpoint used for normal data transfers (bits 5..4 = 00b), whether it is used to convey explicit feedback information for one or more data endpoints (bits 5..4 = 01b) or whether it is a data endpoint that also serves as an implicit feedback endpoint for one or more data endpoints (bits 5..4=10b).

If the endpoint is used as an explicit feedback endpoint (bits 5..4 = 01b), then the Transfer Type shall be set to isochronous (bits 1..0 = 01b) and the Synchronization Type shall be set to No Synchronization (bits 3..2 = 00b).

A feedback endpoint (explicit or implicit) needs to be associated with one (or more) isochronous data endpoints to which it provides feedback service. The association is based on endpoint number matching. A feedback endpoint always has the opposite direction from the data endpoint(s) it services. If multiple data endpoints are to be serviced by the same feedback endpoint, the data endpoints shall have ascending ordered–but not necessarily consecutive–endpoint numbers. The first data endpoint and the feedback endpoint shall have the same endpoint number (and opposite direction). This ensures that a data endpoint can uniquely identify its feedback endpoint by searching for the first feedback endpoint that has an endpoint number equal or less than its own endpoint number.

Example: Consider the extreme case where there is a need for five groups of OUT asynchronous isochronous endpoints and at the same time four groups of IN adaptive isochronous endpoints. Each group needs a separate feedback endpoint and the groups are composed as shown in Table 9-21.

Table 9-21. Example of Feedback Endpoint Numbers

OUT Group	Number of OUT Endpoints	IN Group	Number of IN Endpoints
1	1	6	1
2	2	7	2
3	2	8	3
4	3	9	4
5	3		

The endpoint numbers can be intertwined as illustrated in Figure 9-7.

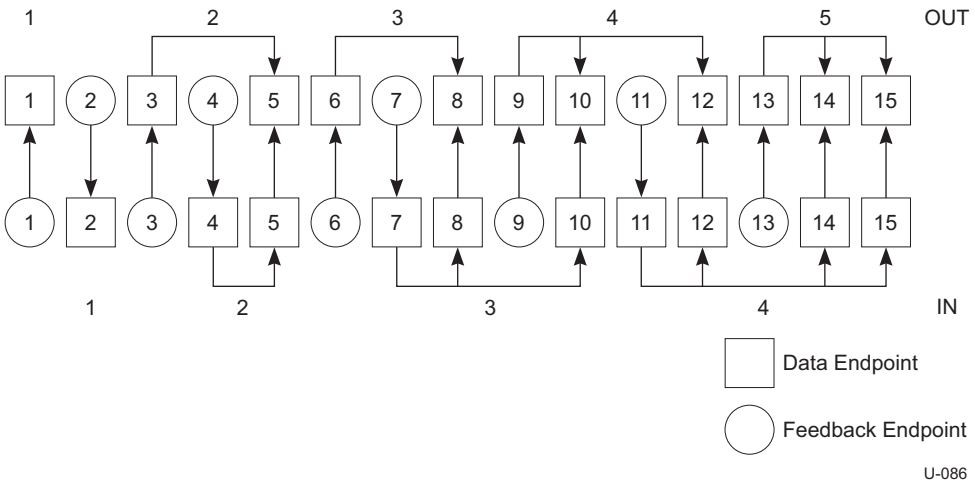


Figure 9-7. Example of Feedback Endpoint Relationships

For high-speed bulk and control OUT endpoints, the *bInterval* field is only used for compliance purposes; the host controller is not required to change its behavior based on the value in this field.

9.6.7 SuperSpeed Endpoint Companion

Each SuperSpeed endpoint described in an interface is followed by a SuperSpeed Endpoint Companion descriptor. This descriptor contains additional endpoint characteristics that are only defined for SuperSpeed endpoints. This descriptor is always returned as part of the configuration information returned by a GetDescriptor(Configuration) request and cannot be directly accessed with a GetDescriptor() or SetDescriptor() request. The Default Control Pipe does not have an Endpoint Companion descriptor. The Endpoint Companion descriptor shall immediately follow the endpoint descriptor it is associated with in the configuration information.

Table 9-22. SuperSpeed Endpoint Companion Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	SUPERSPEED_USB_ENDPOINT_COMPANION Descriptor Type
2	*bMaxBurst*	1	Number	The maximum number of packets the endpoint can send or receive as part of a burst. Valid values are from 0 to 15. A value of 0 indicates that the endpoint can only burst one packet at a time and a value of 15 indicates that the endpoint can burst up to 16 packets at a time. For endpoints of type control this shall be set to 0.

Offset	Field	Size	Value	Description
3	*bmAttributes*	1	Bitmap	If this is a Bulk Endpoint: Bits Description 4:0 MaxStreams. The maximum number of streams this endpoint supports. Valid values are from 0 to 16, where a value of 0 indicates that the endpoint does not define streams. For the values 1 to 16, the number of streams supported equals $2^{MaxStream}$. 7:5 Reserved. These bits are reserved and shall be set to zero. If this is a Control or Interrupt Endpoint: Bits Description 7:0 Reserved. These bits are reserved and shall be set to zero. If this is an isochronous endpoint: Bits Description 1:0 Mult. A zero based value that determines the maximum number of packets within a service interval that this endpoint supports. Maximum number of packets = (bMaxBurst + 1) x (Mult + 1) The maximum value that can be set in this field is 2. This field shall be set to zero if the bMaxBurst field is set to zero. 7:2 Reserved. These bits are reserved and shall be set to zero.
4	*wBytesPerInterval*	2	Number	The total number of bytes this endpoint will transfer every service interval. This field is only valid for periodic endpoints. For isochronous endpoints, this value is used to reserve the bus time in the schedule, required for the frame data payloads per 125 µs. The pipe may, on an ongoing basis, actually use less bandwidth than that reserved. The device reports, if necessary, the actual bandwidth used via its normal, non-USB defined mechanisms. wBytesPerInterval is reserved and must be set to zero for control and bulk endpoints.

9.6.8 String

String descriptors are optional. As noted previously, if a device does not support string descriptors, all references to string descriptors within device, configuration, and interface descriptors shall be reset to zero.

String descriptors use UNICODE UTF16LE encodings as defined by *The Unicode Standard, Worldwide Character Encoding, Version 5.0*, The Unicode Consortium, Addison-Wesley Publishing Company, Reading, Massachusetts (http://www.unicode.org). The strings in a device may support multiple languages. When requesting a string descriptor, the requester specifies the desired language using a 16-bit language ID (LANGID) defined by the USB-IF. The list of currently defined USB LANGIDs can be found at http://www.usb.org/developers/docs.html. String index zero for all languages returns a string descriptor that contains an array of 2-byte LANGID codes supported by the device. Table 9-23 shows the LANGID code array. A device may omit all string descriptors. Devices that omit all string descriptors shall not return an array of LANGID codes.

The array of LANGID codes is not NULL-terminated. The size of the array (in bytes) is computed by subtracting two from the value of the first byte of the descriptor.

Table 9-23. String Descriptor Zero, Specifying Languages Supported by the Device

Offset	Field	Size	Value	Description
0	*bLength*	1	N+2	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	STRING Descriptor Type
2	*wLANGID[0]*	2	Number	LANGID code zero
...	...	...	...	...
N	*wLANGID[x]*	2	Number	LANGID code *x*

The UNICODE string descriptor (shown in Table 9-24) is not NULL-terminated. The string length is computed by subtracting two from the value of the first byte of the descriptor.

Table 9-24. UNICODE String Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	STRING Descriptor Type
2	*bString*	N	Number	UNICODE encoded string

9.7 Device Class Definitions

All devices shall support the requests and descriptor definitions described in this chapter. Most devices provide additional requests and, possibly, descriptors for device-specific extensions. In addition, devices may provide extended services that are common to a group of devices. In order to define a class of devices, the following information shall be provided to completely define the appearance and behavior of the device class.

9.7.1 Descriptors

If the class requires any specific definition of the standard descriptors, the class definition shall include those requirements as part of the class definition. In addition, if the class defines a standard extended set of descriptors, they shall also be fully defined in the class definition. Any extended descriptor definitions shall follow the approach used for standard descriptors; for example, all descriptors shall begin with a length field.

9.7.2 Interface(s)

When a class of devices is standardized, the interfaces used by the devices shall be included in the device class definition. Devices may further extend a class definition with proprietary features as long as they meet the base definition of the class.

9.7.3 Requests

All of the requests specific to the class shall be defined.

10 Hub, Host Downstream Port, and Device Upstream Port Specification

This chapter describes the architectural requirements for a USB 3.0 hub. The chapter also describes differences between functional requirements for a host downstream port and a hub downstream port as well as differences between a device upstream port and a hub upstream port. The chapter contains descriptions of two of the three principal sub-blocks: the SuperSpeed hub repeater/forwarder and the SuperSpeed hub controller. The USB 2.0 hub sub-block is described in the *Universal Serial Bus Specification, Revision 2.0*. This chapter also describes the hub's operation for error recovery, reset, suspend/resume, hub request behavior, and hub descriptors.

The hub specification chapter along with the *Universal Serial Bus Specification, Revision 2.0* supply the information needed for an implementer to design a hub that conforms to the USB 3.0 specification.

10.1 Hub Feature Summary

Hubs provide the electrical interface between USB devices and the host. Hubs are directly responsible for supporting many of the attributes that make USB user friendly and hide its complexity from the user. Listed below are the major aspects of USB functionality that hubs support:

- Connectivity behavior
- Power management
- Device connect/disconnect detection
- Bus fault detection and recovery
- SuperSpeed and USB 2.0 (high-speed, full-speed, and low-speed) device support

A USB 3.0 hub incorporates a USB 2.0 hub and a SuperSpeed hub consisting of two principal components: the SuperSpeed Hub Repeater/Forwarder and the SuperSpeed Hub Controller. The USB 2.0 hub is described in the USB 2.0 specification. All subsequent references in this specification are to components of the SuperSpeed hub unless otherwise noted. The Hub Repeater/Forwarder is responsible for connectivity setup and tear-down. It also supports exception handling, such as bus fault detection and recovery and connect/disconnect detect. The Hub Controller provides the mechanism for host-to-hub communication. Hub-specific status and control commands permit the host to configure a hub and to monitor and control its individual downstream facing ports.

Figure 10-2 shows a high level block diagram of a four port USB 3.0 hub and the locations of its upstream and downstream facing ports. A USB 3.0 hub is the logical combination of two hubs: a USB 2.0 hub and a SuperSpeed hub. Each hub operates independently on a separate data bus. Typically, the only signal shared logic between them is to control VBUS. If either the USB 2.0 hub or SuperSpeed hub controllers requires, a downstream port is powered. A USB 3.0 hub connects on both interfaces upstream whenever possible. All exposed downstream ports on a USB 3.0 hub shall support both SuperSpeed and USB 2.0 connections. Host controller ports may have different requirements.

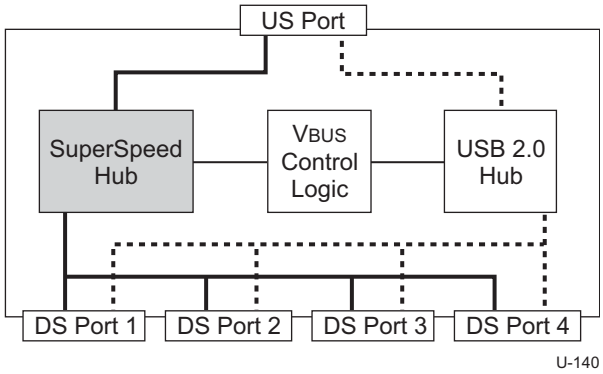


Figure 10-1. Hub Architecture

Figure 10-2 shows the SuperSpeed portion of a USB 3.0 hub consisting of a Hub Repeater/Forwarder section and a Hub Controller section.

The USB 2.0 portion of a USB 3.0 hub shall meet all requirements of the USB 2.0 specification unless specific exceptions are noted.

The Hub Repeater/Forwarder is responsible for managing connectivity between upstream and downstream facing ports which are operating at SuperSpeed. The Hub Controller provides status and control and permits host access to the SuperSpeed Hub.

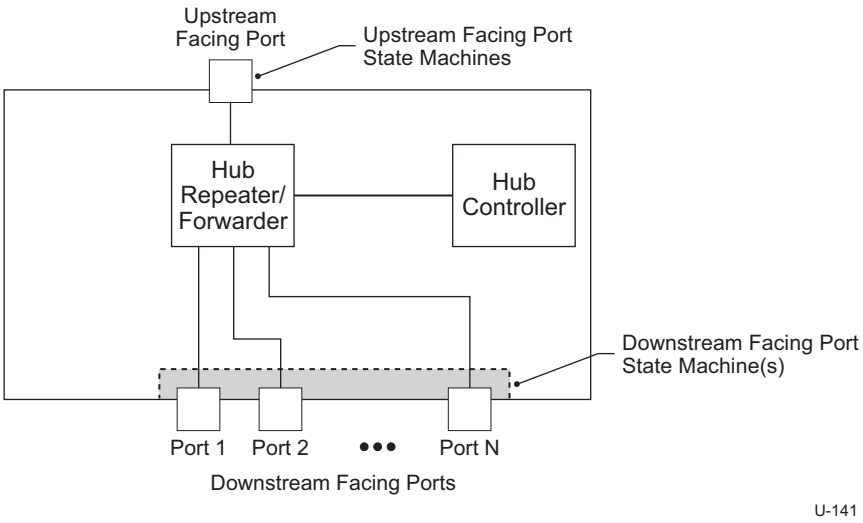


Figure 10-2. SuperSpeed Portion of the Hub Architecture

When the hub upstream facing port is attached to an electrical environment that is only operating at high-speed or full-speed, SuperSpeed connectivity is not available to devices attached to downstream facing ports.

Unlike USB 3.0 peripheral devices, a USB 3.0 hub is required to connect upstream on both the USB 3.0 and USB 2.0 buses. SuperSpeed connections may be enabled or disabled under the control of system software for a USB 3.0 hub's downstream ports. If the hub upstream SuperSpeed connection is not supported by the port to which the USB 3.0 hub is connected, the hub disables SuperSpeed support on all of its downstream ports. If a USB 3.0 hub upstream port is not connected on either USB 2.0 or SuperSpeed, the hub does not provide power to the downstream ports unless it supports the USB Implementers Forum, Inc.'s Battery Charging Specification. Refer to Section 10.3.1.1 for a detailed discussion on when a hub is allowed to remove VBUS from a downstream facing port. The USB 3.0 specification allows self-powered and bus-powered hubs.

The following sections present the typical flow for connection management in various types of systems for the simple topology shown in Figure 10-3 when the host system is first powered on.

Note: These connection examples outline cases where the system operates as expected. The handling of error cases are specified later in this chapter.

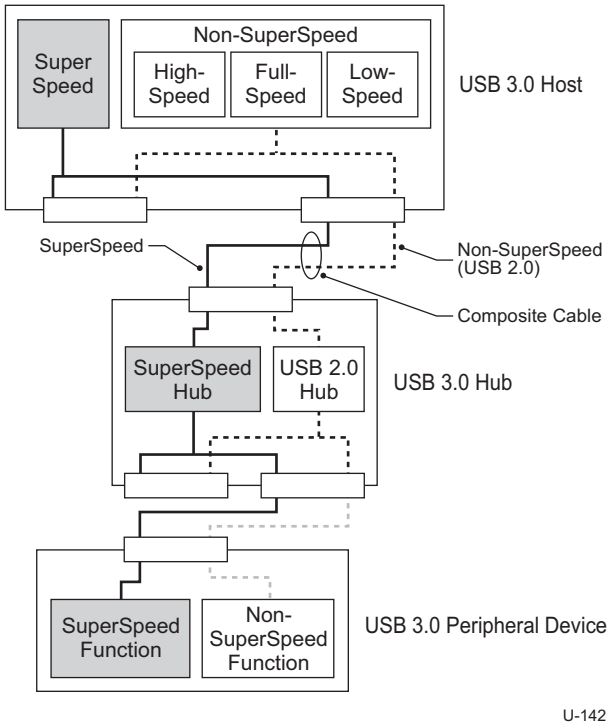


Figure 10-3. Simple USB 3.0 Topology

10.1.1 SuperSpeed Capable Host with SuperSpeed Capable Software

When the host is powered off, the hub does not provide power to its downstream ports unless the hub supports charging applications (refer to Section 10.3.1.1).

When the host is powered on with SuperSpeed support enabled on its downstream ports by default the following is the typical sequence of events:

- Hub detects VBUS and SuperSpeed support and powers its downstream ports with SuperSpeed enabled.
- Hub connects both as a SuperSpeed and as a high-speed device.
- Device detects VBUS and SuperSpeed support and connects as a SuperSpeed device.
- Host system begins hub enumeration at high-speed and SuperSpeed.
- Host system begins device enumeration at SuperSpeed.

10.1.2 USB 2.0 Host

When the host is powered off, the hub does not provide power to its downstream ports unless the hub supports charging applications (refer to Section 10.3.1.1).

When the host is powered on and there is no SuperSpeed hardware support, the following is the typical sequence of events:

- Hub detects VBUS and connects as a high-speed device.
- Host system begins hub enumeration at high-speed.

- Hubs power downstream ports when directed by software (USB 2.0) with SuperSpeed support disabled.
- Device connects at high-speed.
- Host system begins device enumeration at high-speed.

10.1.3 Hub Connectivity

Hubs exhibit different connectivity behavior depending on whether they are propagating data packet header/data packet payload packet traffic, other packet traffic, resume signaling, or are in an Idle state.

10.1.3.1 Packet Signaling Connectivity

The Hub Repeater/Forwarder contains one port that shall always connect in the upstream direction (referred to as the upstream facing port) and one or more downstream facing ports. Upstream connectivity is defined as being towards the host, and downstream connectivity is defined as being towards a device. A SuperSpeed hub controller contains buffering for header and data packets. A SuperSpeed hub controller does not use the repeater-only model used for high-speed connectivity in a USB 2.0 hub. This change allows multiple downstream devices to send asynchronous messages simultaneously without data loss and for some traffic to be stored and delivered when it is directed to downstream ports when the links are not in U0.

Figure 10-4 shows the high level packet signaling connectivity behavior for hubs in the upstream and downstream directions. Later sections describe the hub internal buffering and connectivity in more detail. A hub also has an Idle state, during which the hub makes no connectivity. When in the Idle state, all of the hub's ports (upstream plus downstream) are U1, U2 or in U0 receiving and transmitting logical idles waiting for the start of the next packet.

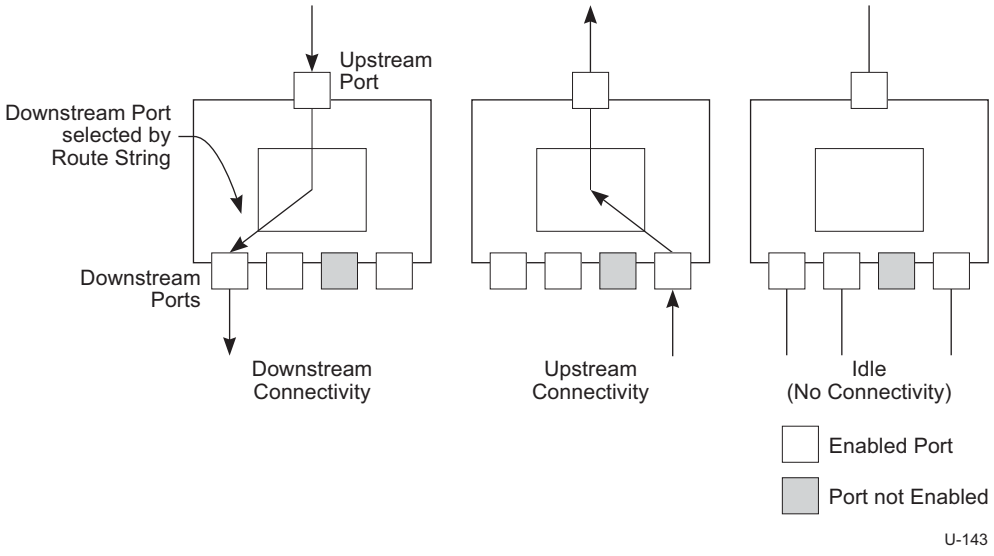


Figure 10-4. Hub Signaling Connectivity

If a downstream facing port is enabled (i.e., in a state where it can propagate signaling through the hub) and the hub detects the start of a packet on that port, the hub begins to store the packet header. Connectivity is established in an upstream direction to the upstream facing port of that hub whenever a valid header packet has been received on a downstream port. The hub transmits the valid header packet received on the downstream port upstream, but not to any other downstream facing ports. This means that when a device or a hub transmits a packet upstream, only those hubs in a direct line between the transmitting device and the host will see the packet.

All packets except Isochronous Timestamp Packets are unicast in the downstream direction; hubs operate using a direct connectivity model. This means that when the host or hub transmits a packet downstream, only those hubs in a direct line between the host and recipient device will see the packet. When a hub detects the start of a packet on its upstream facing port, the hub begins to store the packet header. Whenever a valid header packet has been received on a hub upstream port, the hub uses the Route String in the packet header packet and the hub depth value assigned when enumerated to establish connectivity only to the port indicated. If the indicated port is not enabled, it does not propagate packet signaling downstream. The hub shall silently drop a header packet that is routed to a downstream port that is not enabled, a port with a link in U3, or a downstream port that does not exist. The hub shall perform normal link level acknowledgement of the header packet in these cases.

10.1.3.2 Routing Information

Packets received on the hub upstream port are routed based on information contained in a 20-bit field (Route String) in the packet header. The route string is used in conjunction with a hub depth value by the hub to identify the target port for a downstream directed packet. The hub depth value is assigned by software using the Set Hub Depth request. The hub ignores the route string and assumes all packets are routed directly to the hub until it enters the configured state and the hub's depth is set. The hub's upstream port shall be represented by port number zero while the downstream ports shall begin with port number one and count up sequentially. Whenever a hub controller responds to a packet routed to the hub with a packet containing a route string or originates a packet (except for a packet the hub is deferring) the hub shall set the route string to zero.

Figure 10-5 illustrates the use of route strings in an example topology with five levels of four port USB 3.0 hubs. The hub depth value for each level of hub is illustrated in the figure. Each hub and each device in the topology contains the route string that would be used to route a packet to that device/hub. For each hub depth, the octet in the route string that determines the routing target at that hub depth is shown in bold and a larger font size than the rest of the route string. The host root port is not included in the 20-bit route string.

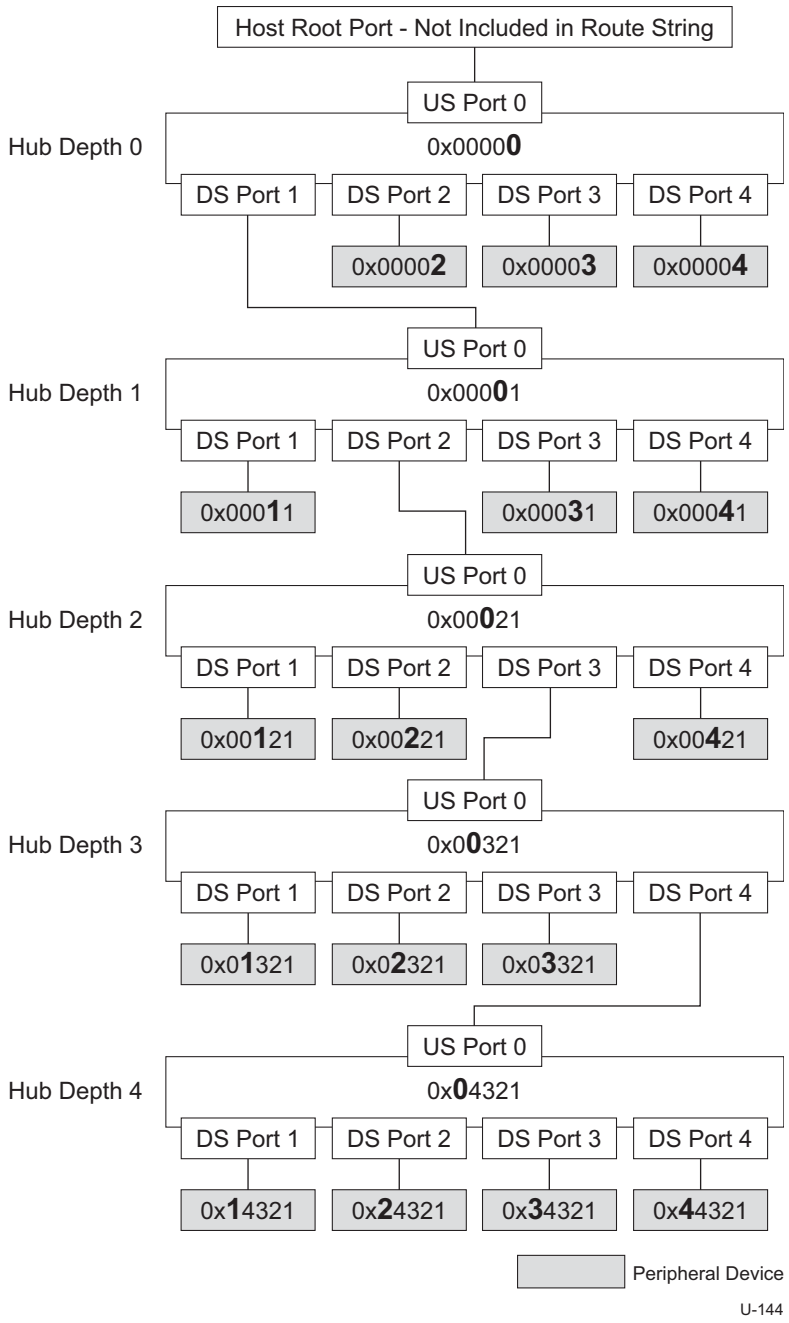


Figure 10-5. Route String Example

10.1.4 Resume Connectivity

Hubs exhibit different connectivity behaviors for upstream- and downstream-directed resume signaling. A hub does not propagate resume signaling from its upstream facing port to any of its downstream facing ports unless a downstream facing port is suspended and has received resume signaling since it was suspended. Figure 10-6 illustrates hub upstream and downstream resume connectivity.

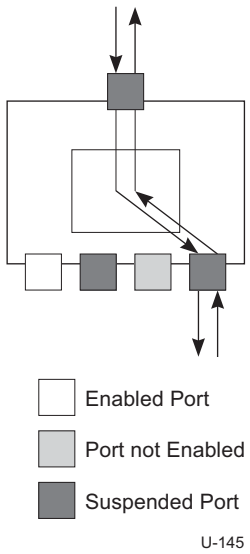


Figure 10-6. Resume Connectivity

If a hub upstream port is suspended and the hub detects resume signaling from a suspended downstream facing port, the hub propagates that signaling upstream and does not reflect that signaling to any of the downstream facing ports (including the downstream port that originated resume signaling). If a hub upstream port is not suspended and the hub detects resume signaling from a suspended downstream facing port, the hub reflects resume signaling to the downstream port. Note that software shall not initiate a transition to U3 on the upstream port of a hub unless it has already initiated transitions to U3 on all enabled downstream ports. A detailed discussion of resume connectivity appears in Section 10.8.

10.1.5 Hub Fault Recovery Mechanisms

Hubs are the essential USB component for establishing connectivity between the host and other devices. It is vital that any connectivity faults be prevented if possible and detected in the unlikely event they occur.

Hubs must also be able to detect and recover from lost or corrupted packets that are addressed to the Hub Controller. Because the Hub Controller is, in fact, another USB device, it shall adhere to the same rules as other USB devices, as described in Chapter 8.

10.1.6 Hub Header Packet Buffer Architecture

Figure 10-7 shows the logical representation of a typical header packet buffer implementation for a SuperSpeed hub. Logically, a SuperSpeed hub has separate header packet buffers associated with each port for both upstream and downstream traffic. When a hub receives a header packet on its upstream port, it routes the header packet to the appropriate downstream header packet buffer for transmission (unless the header packet is for the hub). When the hub receives a non-LMP header packet on a downstream port, it routes the header packet to the upstream port header packet buffer for transmission. Header packets are kept in the hub header packet buffers after transmission until link level acknowledgement (LGOOD_n) for the header packet is received. This allows the hub to retry the header packets if necessary to ensure that header packets are received correctly at the link level. The header packet buffers also allow a hub to store the header packets until they can be forwarded when the header packet is directed to a downstream link that is a low power link state. Hubs store the header packet and deliver it once the link becomes active.

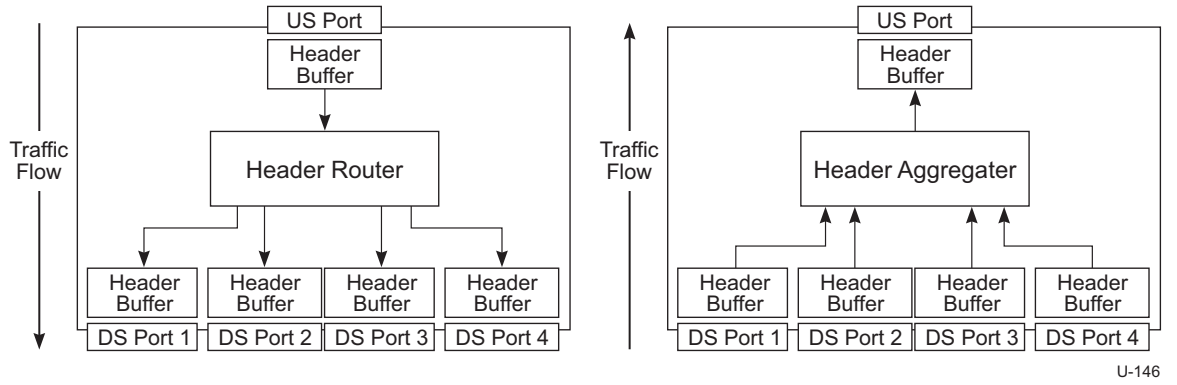


Figure 10-7. Typical Hub Header Packet Buffer Architecture

10.1.6.1 Hub Data Buffer Architecture

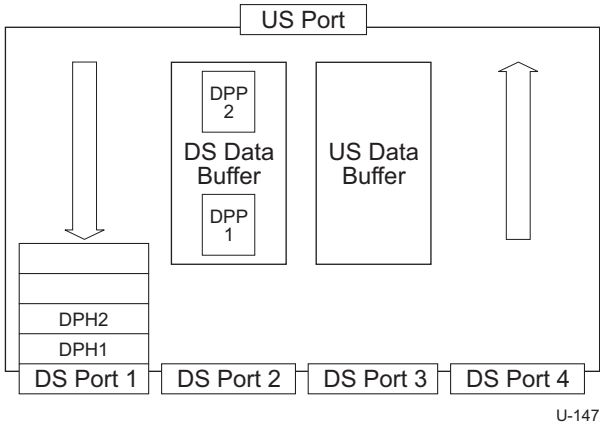


Figure 10-8. Hub Data Buffer Traffic (Header Packet Buffer Only Shown for DS Port 1)

Figure 10-8 shows the logical representation of the data buffer architecture in a typical SuperSpeed hub. SuperSpeed hubs provide independent buffering for data packet payloads (DPP) in both the upstream and downstream directions. The USB 3.0 Architecture allows concurrent transactions to occur in both the upstream and downstream directions. In the figure, two data packets are in progress in the downstream direction. The hub can store more than one data packet payload at the same time. In rare occurrences where data packet payloads are discarded because buffering is unavailable, the end-to-end protocol will recover by retrying the transaction. The isochronous protocol does not include retries. However, discard errors are expected less frequently then bit errors on the physical bus.

Note: Data packet headers are stored and handled in the same fashion as other header packet packets using the header packet buffers. DPPs are handled using the separate data buffers.

10.2 Hub Power Management

10.2.1 Link States

The hub is required to support U0, U1, U2, and U3 on all ports (upstream and downstream).

10.2.2 Hub Downstream Port U1/U2 Timers

The hub is required to have inactivity timers for both U1 and U2 on each downstream port. The timeout values are programmable and may be set by the host software. A timeout value of zero means the timer is disabled. The default value for the U1/U2 timeouts is zero. The U1 and U2 timeout values for all downstream ports reset to the default state on PowerOn Reset or when the hub upstream port is reset. The U1 and U2 timeout values for a downstream port reset to the default values when the port receives a SetPortFeature request for a port reset. The downstream port state machines presented in this chapter describe the specific operational rules when U1 and/or U2 timeouts are enabled.

- Hub downstream ports shall accept U1 or U2 entry initiated by a link partner except when the corresponding U1/U2 timeout is set to zero or there is pending traffic directed to the downstream port.
- If a hub has received a valid packet on its upstream port that is routed to a downstream port, it shall reject U1 or U2 link entry attempts on the downstream port until the packet has been successfully transmitted. A hub may also reject U1 or U2 link entry attempts on downstream ports if the hub is receiving a packet but has not determined the packet's destination. A hub implementation shall ensure no race condition exists where a header packet that has not been deferred is queued for transmission on a downstream port with a link that is in U1, U2, or is in the process of entering U1, U2.
- Hub downstream ports shall reject all U1 and U2 entry requests if the corresponding timeout is set to zero.
- The hub inactivity timers for U1 and U2 shall not be reset by an Isochronous Timestamp Packet.

10.2.3 Downstream/Upstream Port Link State Transitions

The hub shall evaluate the link power state of its downstream ports such that it propagates the highest link state of any of its downstream ports to its upstream port when there is no pending upstream traffic. U0 is the highest link state, followed by U1, then U2, then U3, then Rx.Detect, and then SS.Disabled. The order of the other link states is undefined and implementation dependent. If an upstream port link state transition would result in an upstream port link state that has been disabled by software, the hub shall transition the upstream port link to the next highest U-state that is enabled. The hub never automatically attempts to transition the hub upstream port to U3 or lower state.

The downstream port state machines presented in this chapter provide the specific timing requirements for changing the upstream port link state in response to downstream port link state changes.

The hub also shall initiate a link state transition on the appropriate downstream port whenever it receives a packet that is routed to downstream port that is not in U0. The hub upstream port state machines provided in this chapter provide the specific timing requirements for these transitions.

If enabled, port status change interrupts, e.g., due to a connect event on a downstream port, will cause the upstream link to initiate a transition to U0.

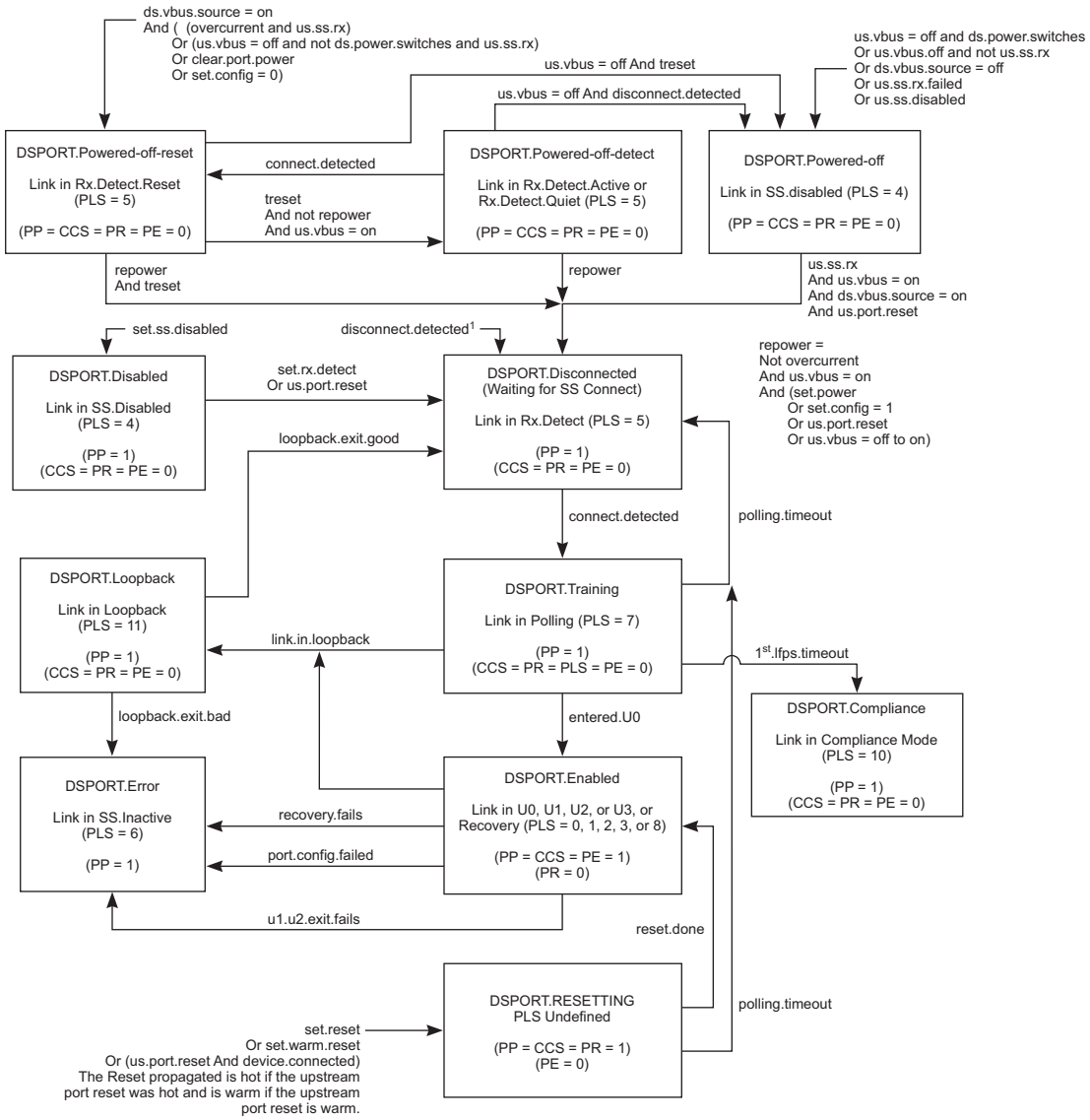
10.3 Hub Downstream Facing Ports

The following sections provide a functional description of a state machine that exhibits the correct required behavior for a downstream facing port.

Figure 10-9 is an illustration of the downstream facing port state machine. Each of the states is described in Section 10.4.2. In the diagram below, some of the entry conditions into states are shown without origin. These conditions have multiple origin states and the individual transitions lines are not shown to simplify the diagram. The description of the entered state indicates from which states the transition is applicable.

NOTE

For the root hub, the signals from the upstream facing port state machines are implementation dependent.



Port Status Field

Notation	Field Name
PP	PORT_POWER
CCS	PORT_CONNECTION
PR	PORT_RESET
PLS	PORT_LINK_STATE
PE	PORT_ENABLE

Notes:
1. GetPortStatus Requests will return all zeros until a Set Configuration Request for a non-zero config and a subsequent Set Port Feature (PORT_POWER) request have occurred for the port.
2. Clear Port Feature (PORT_ENABLED) and Set Port Feature (PORT_ENABLED) are not used for SS Ports.

[1] This direct transition may only occur from a DSPORT state whose link is in the SS.Inactive, Rx.Detect.Active (during DSPORT.RESETTING), U1, U2, or U3 state.

U-148A

Figure 10-9. Downstream Facing Hub Port State Machine

Table 10-1. Downstream Facing Hub Port State Machine Diagram Legend

Key	Description
1st.lfps.timeout	First LFPS Timeout
clear.port.power	Received ClearPortFeature(PORT_POWER)
connect.detected	Connect Detected
device.connected	Device is connected
disconnect.detected	Disconnect Detected
ds.power.switches	Hub reports non-zero value in bPwrOn2PwrGood field of hub descriptor. Hub supports power switching.
ds.vbus.source = off	Downstream port VBUS is Off due to loss of Local Power Source when hub is self-powered.
ds.vbus.source = on	Downstream port VBUS may be On. Local Power Source is On or the hub is bus-powered.
entered.U0	Link Transitions from Polling.Idle to U0
link.in.loopback	Loopback bit set in received TS2 ordered sets
loopback.exit.bad	Loopback exit LFPS handshake failed (applies only if Downstream Port is loopback master)
loopback.exit.good	Loopback exit LFPS handshake successful
overcurrent	Over-Current is active.
polling.timeout	Any Polling substate times out
port.config.failed	Port Configuration Fails (refer to Section 8.4.5)
recovery.fails	Link Exits Recovery after Timeout
repower	Repowering conditions defined in 10.3.1.10 are met
reset.done	Reset Completes Successfully
set.config = 0	Received SetConfig(0) request
set.config = 1	Received SetConfig(1) request
set.port.power	Received SetPortFeature(PORT_POWER)
set.reset	Received SetPortFeature(PORT_RESET)
set.rx.detect	Received SetPortFeature(PORT_LINK_STATE) = Rx.Detect
set.ss.disabled	Received SetPortFeature(PORT_LINK_STATE) = SS.Disabled
set.warm.reset	Received SetPortFeature(PORT_BH_RESET)
treset	Warm Reset has been signaled for tReset duration.
us.port.reset	Received inband reset on Upstream Port
us.ss.disabled	Upstream port transitioned to SS.Disabled.
us.ss.rx	Far End Receiver Terminations are present on the Upstream Port's link or were present when Upstream VBUS was most recently on.
us.ss.rx.failed	Hub's upstream port link has attempted eight consecutive Rx.Detect events without detecting far-end receiver termination
us.vbus = off	Upstream port VBUS is Off
us.vbus = on	Upstream port VBUS is On
u1.u2.exit.fails	U1 or U2 Exit Fails

10.3.1 Hub Downstream Facing Port State Descriptions

10.3.1.1 DSPORT.Powered-off

The DSPORT.Powered-off state is a logical powered off state. The hub may still be required or choose to provide VBUS for a downstream port in the DSPORT.Powered-off state. Detailed requirements for presence of VBUS are covered later in this section.

A port shall transition into this state if any of the following situations occur:

- From any state when local power is lost to the port.
- From any state when VBUS is removed from the hub upstream port and the hub supports power switching on the DS ports.
- From any state if the hub's upstream port link transitions to the SS.Disabled state and Upstream PORT VBUS is on.
- From any state if the hub's upstream port link has attempted eight consecutive Rx.Detect events without detecting far-end receiver terminations.
- From the DSPORT.Powered-off-detect state if the hub's Upstream Port Vbus is Off, the hub does not support power switching and far-end receiver detection has failed on the downstream port and the conditions for Repowering defined in Section 10.3.1.10 are not met.
- From the DSPORT.Powered-off-reset state if the hub's Upstream Port Vbus is Off, the hub does not support power switching, conditions for Repowering defined in Section 10.3.1.10 are not met and the DSPORT.Powered-off-reset state has been maintained for tReset.

A port shall remain in the DSPORT.Powered-off state until the following conditions are met.

- The hub's Upstream Port link has detected far-end receiver terminations. Note that this requires Upstream Port VBUS to be on.
- Power is available for the downstream port. This is true if the hub is bus-powered or the local power source is on.

If a hub was configured while the local power supply was present and then if local power is lost, the hub shall place all ports in the Powered-off state if power remains to run the hub controller.

In the DSPORT.Powered-off state, the port's link is in the SS.Disabled state.

Table 10-2 shows the allowed state of VBUS for hub downstream ports for possible states of the hub upstream port and logical port power for a downstream port. The table covers the case where the hub has adequate power to provide power for the downstream ports (local power source is present). For a hub that does not implement per port power control, all downstream ports that will be affected by removing VBUS shall be in a state where power may be off (refer to Table 10-2) before the hub removes VBUS.

Note: a hub may provide power to all its downstream ports all of the time to support applications such as battery charging from a USB port.

Table 10-2. Downstream Port VBUS Requirements

Hub Upstream Port Connection Status	Downstream Port SuperSpeed Port Power Off (PORT_POWER = 0) USB 2.0 Port Power On (PORT_POWER = 1)	Downstream Port SuperSpeed Port Power On (PORT_POWER = 1) Downstream Port USB 2.0 Port Power Off (PORT_POWER = 0)	Downstream Port USB 2.0 and SuperSpeed Port Power Off (PORT_POWER = 0)
SuperSpeed	On*	On	May be off
USB 2.0	On	May be off	May be off
SuperSpeed and USB 2.0	On	On	May be off
No VBUS	May be off	May be off	May be off

* If the hub upstream port is unable to connect on the USB 2.0 bus, the downstream port VBUS may be off in this state.

10.3.1.2 DSPORT.Disconnected (Waiting for SS Connect)

This is the default state when local power is valid (self-powered) or VBUS becomes valid (bus-powered). A port transitions to this state in any of the following situations:

- From the DSPORT.Powered-off state when the hub's Upstream Port link has detected far-end receiver terminations, Upstream Port VBUS is on (implied by receiver detection) and power is available for the downstream port. Power is available for the downstream port if the hub is bus-powered or if the local power source is on.
- From any state that can and does detect a disconnect, except from DSPORT.Powered-off-detect. For a port to detect a disconnect, it must be in a state for which its current connect status, CCS, is 1.
- From the DSPORT.Powered-off-reset state when conditions for Repowering defined in Section 10.3.1.10 are met and the DSPORT.Powered-off-reset state has been maintained for tReset.
- From the DSPORT.Powered-off-detect state when conditions for Repowering defined in Section 10.3.1.10 are met.
- From the DSPORT.Resetting state when a port's link times out from Rx.Detect.Active during a reset. That is, it detects a disconnect.
- From the DSPORT.Disabled state when a SetPortFeature(PORT_LINK_STATE) Rx.Detect request is received for the port.
- From the DSPORT.Powered-off state or DSPORT.Disabled state when the hub's upstream port is reset. Note: The hub shall issue a Warm Reset on the downstream port after it has transitioned the port to the DSPORT.RxDetect state and detected a far-end receiver, even if the upstream port reset is a hot reset
- From the DSPORT.Resetting state if the port's link times out from any Polling substate during a reset.
- From the DSPORT.Training state if the port's link times out from any Polling substate.
- From the DSPORT.Loopback state if the port's link performs a successful LFPS handshake in Loopback.Exit.

In this state, the port's link shall be in the Rx.Detect state.

Note: The port's link shall still perform connection detection normally from the Rx.Detect if the hub upstream port's link is in U3.

10.3.1.3 DSPORT.Training

A port transitions to this state from the DSPORT.Disconnected state when SuperSpeed far-end receiver terminations are detected.

In this state, the port's link shall be in the Polling state.

10.3.1.4 DSPORT.ERROR

A port shall transition to this state only when a SuperSpeed capable device is present and a serious error condition occurred while attempting to operate the link.

A port transitions to this state in any of the following situations:

- From the DSPORT.Enabled state if the link enters recovery and times out without recovering.
- From the DSPORT.Resetting state if U1 or U2 exit fails.
- From the DSPORT.Loopback state if the port is the loopback master and the LFPS handshake in Loopback.Exit fails.
- From DSPORT.Enabled if Port Configuration fails as described in Section 8.4.5.

In this state, the port's link shall be in the SS.Inactive state.

10.3.1.5 DSPORT.Enabled

A port transitions to this state in any of the following situations:

- From the Training state when the port's link successfully enters U0.
- From the DSPORT.Resetting state when a reset completes successfully.

A port in the DSPORT.Enabled state will propagate packets in both the upstream and the downstream direction. When the hub downstream port first transitions to the DSPORT.Enabled state after a power on or warm reset, it shall transmit a port configuration LMP as defined in Section 8.4.5.

When the hub downstream port first transitions to the DSPORT.Enabled state after a power on reset, the value for the U1 and U2 inactivity timers shall be reset to zero.

The link shall be in U0 when the enabled state is entered.

If the hub upstream port's link is in U3 when the downstream port enters DSPORT.Enabled and the hub is not enabled for remote wakeup, the downstream port shall initiate a transition to U3 on its link within tDSPortEnabledToU3.

Section 10.4 provides a state machine that shows a functionally correct implementation for a downstream port managing different link states within the DSPORT.Enabled state.

10.3.1.6 DSPORT.Resetting

A downstream port transitions to the DSPORT.Resetting state in any of the following situations:

- From the DSPORT.Error state when a SetPortFeature(PORT_RESET) request or SetPortFeature(BH_PORT_RESET) is received, the port shall send a warm reset on the downstream port link.
- From the DSPORT.Enabled state and the port's link is in any state when a SetPortFeature(BH_PORT_RESET) is received. In this situation the port shall initiate a Warm Reset on the downstream port link.

- From any state except for DSPORT.Powered-off-reset or DSPORT.Powered-off-detect or DSPORT.Powered-off or DSPORT.Disabled or DSPORT.Disconnected if the hub detects a Reset on its Upstream Port. In this situation, the port shall initiate a Hot/Warm Reset on the downstream port link depending on the type of Reset detected on the hub's upstream port and depending on the current state of the downstream port. This transition shall occur before the upstream port link transitions to U0.
- From any state except for DSPORT.Powered-off or DSPORT.Disconnected when it receives a SetPortFeature(PORT_RESET) or SetPortFeature(BH_PORT_RESET). If the downstream port is in the DSPORT.Powered-off or DSPORT.Disconnected state and it receives one of the above requests, the request is ignored.
- From the DSPORT.Enabled state and the port's link state is in any state other than U3 when a SetPortFeature(PORT_RESET) is received. In this situation the port shall initiate a Hot Reset on the downstream port link.
- From the DSPORT.Enabled state and the port's link state is in U3 when a SetPortFeature(PORT_RESET) is received. In this situation the port shall initiate a Warm Reset on the downstream port link.

Note: If the port initiates a hot reset on the link and the hot reset fails during the link Recovery state, a warm reset will be automatically tried. Refer to the Link Chapter for details on this process. The port stays in the DSPORT.Resetting state throughout this process until the warm reset completes.

When the downstream port link enters Rx.Detect.Active during a warm reset, the hub shall start a timer to count the time it is in Rx.Detect.Active or Rx.Detect.Quiet. If this timer exceeds tTimeForResetError while the link remains in Rx.Detect, the port shall transition to the DSPORT.Disconnected state.

10.3.1.7 DSPORT.Compliance

A port transitions to this state in any of the following situations:

- When the link enters the Compliance Mode state.

10.3.1.8 DSPORT.Loopback

A port transitions to this state in any of the following situations:

- From the DSPORT.Training state if the loopback bit is set in the received TS2 ordered sets.

In this state, the port's link shall be in the Loopback state.

10.3.1.9 DSPORT.Disabled

A port transitions to this state when the port receives a SetPortFeature(PORT_LINK_STATE) SS.Disabled request.

In this state, the port's link shall be in the SS.Disabled state.

10.3.1.10 DSPORT.Powered-off-detect

This state is entered when the downstream power state is logically off and a SS, rather than USB 2.0 connection, is desired. To ensure that a SuperSpeed connection is established, unlike the DSPORT.Powered-off state, SS terminations are maintained while in this state. This state shall

perform far-end receiver detection with the link in Rx.Detect, until any of the following conditions are true:

- A receiver is detected.
- The conditions for Repowering the port as described below are met.
- Upstream Port VBUS is off, the hub does not support power switching and receiver detection has failed.

A port shall transition into this state from the DSPORT.Powered-off-reset state when tReset time has been met and the conditions for Repowering are not met.

All the following conditions shall be met for Repowering:

- No overcurrent condition is active.
- Upstream Port VBUS is on.
- SetPortFeature(PORT_POWER) request is received,
  Or SetConfig(1) request is received,
  Or Upstream Port Reset is detected,
  Or Upstream Port VBUS transitioned from off to on.

10.3.1.11 DSPORT.Powered-off-reset

This state is entered when the downstream power state is logically off and a SS, rather than USB 2.0 connection, is desired. To ensure that a SuperSpeed connection is established, unlike the DSPORT.Powered-off state, the SS terminations are maintained while in this state, and to avoid a link training failure, which would allow the downstream device to drop into Compliance Mode or USB 2.0 operation, Warm Reset signaling shall be driven for tReset duration. This state shall drive Warm Reset with the link in the Rx.Detect.Reset substate, until the tReset duration is met.

When any of the following conditions are true, this state is entered regardless of the previous state.

- Overcurrent condition is detected either on this port or globally and Upstream Port Far-end Receiver Terminations are present or were present when Upstream VBUS was last on.
- Upstream Port VBUS is off and hub does not support power switching and Upstream Port Far-end Receiver Terminations were present when Upstream VBUS was last on.
- The hub receives a ClearPortFeature(PORT_POWER) request for this port. In this case, power is removed from the port only if it would not impact the low-speed, full-speed, or high-speed operation on any of the downstream ports on the hub and would not impact SS operation on any ports other than the target port.
- The hub upstream port receives a SetConfiguration(0) request. In this case the downstream port will stay in this state or transition between this state and DSPORT.Powered-off-reset state regardless of other conditions until the hub is reset or the hub upstream port receives a non-zero SetConfiguration request.

10.3.2 Disconnect Detect Mechanism

Disconnect detection mechanisms are covered in Section 7.5.

10.3.3 Labeling

USB system software uses port numbers to reference an individual port with a ClearPortFeature or SetPortFeature request. If a vendor provides a labeling to identify individual downstream facing ports, then each port connector shall be labeled with its respective port number. The port numbers assigned to a specific port by the hub shall be consistent between the USB 2.0 hub and SuperSpeed hub controller.

10.4 Hub Downstream Facing Port Power Management

The following sections provide a functional description of a state machine that exhibits correct link power management behavior for a downstream facing port.

Figure 10-10 is an illustration of the downstream facing port power management state machine. Each of the states is described in Section 10.4.2. In Figure 10-10, some of the entry conditions into states are shown without origin. These conditions have multiple origin states and the individual transitions lines are not shown so that the diagram can be simplified. The description of the entered state indicates from which states the transition is applicable.

10.4.1 Downstream Facing Port PM Timers

Each downstream port maintains logical inactivity timers for keeping track of when U1 and U2 timeouts are exceeded. The U1 or U2 timeout values may be set by software with a SetPortFeature(PORT_U1_TIMEOUT) or SetPortFeature(PORT_U2_TIMEOUT) command at any time. The PM timers are reset to 0 every time a SetPortFeature(PORT_U1_TIMEOUT) or SetPortFeature(PORT_U2_TIMEOUT) request is received. The timers shall be reset every time a packet of any type except an isochronous timestamp packet is sent or received by the port's link. The U1 timer shall be accurate to +1/-0 µs. The U2 timer shall be accurate to +500/-0 µs. Other requirements for the timer are defined in the downstream port PM state machine descriptions.

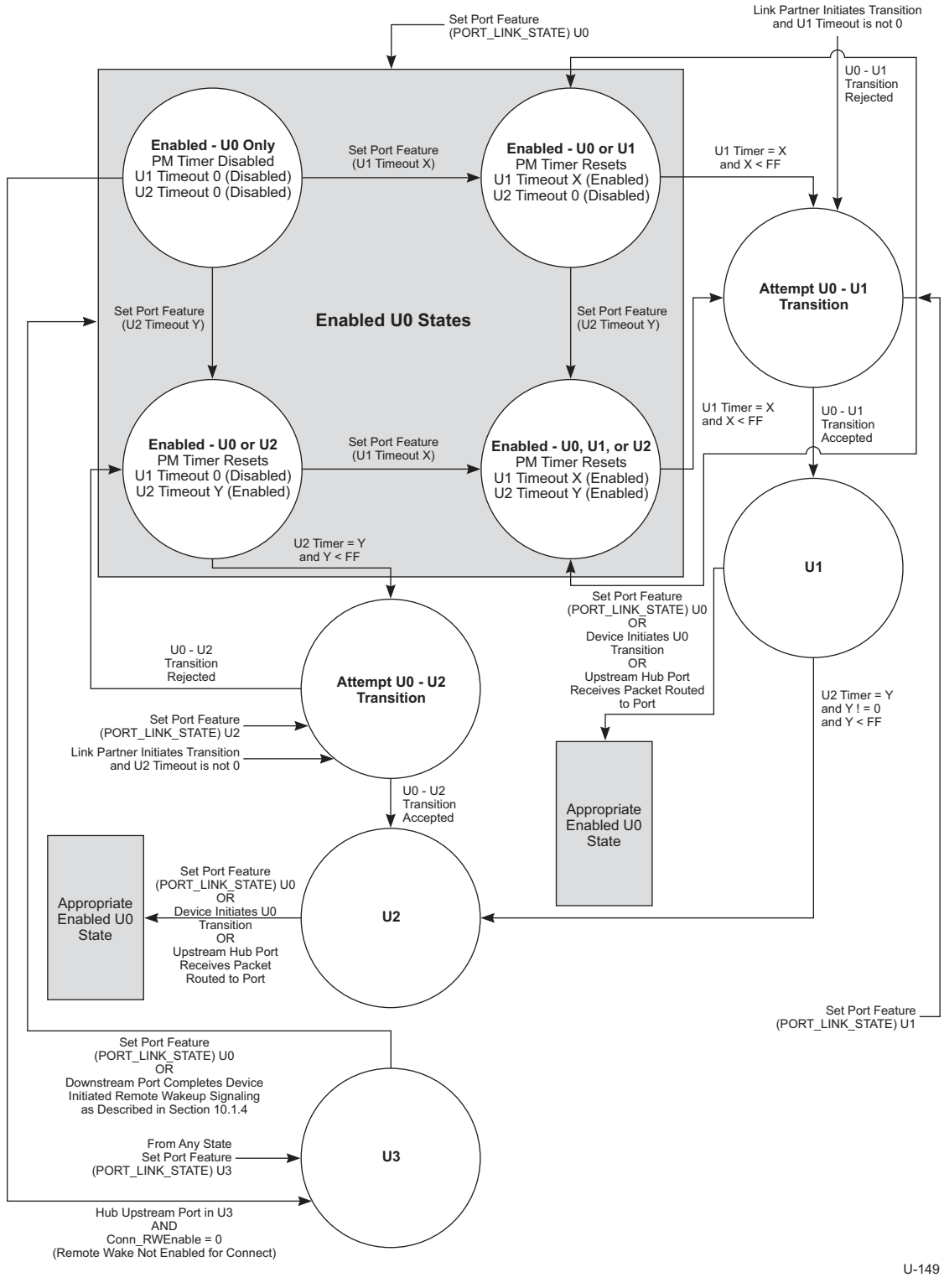


Figure 10-10. Downstream Facing Hub Port Power Management State Machine

10.4.2 Hub Downstream Facing Port State Descriptions

10.4.2.1 Enabled U0 States

There are four enabled U0 states that differ only in the values that are configured for the U1 and U2 timeouts. The port behaves as follows for the various combinations of U1 and U2 timeout values:

U1_TIMEOUT = 0, U2_TIMEOUT = 0

- This is the default state before the hub has received any SetPortFeature(PORT_U1/U2_TIMEOUT) requests for the port.
- The port's link shall reject all U1 or U2 transition requests by the link partner.
- The PM timers may be disabled and the PM timer values shall be ignored.
- The port's link shall not attempt to initiate transitions to U1 or U2.

U1_TIMEOUT = X > 0, U2_TIMEOUT = 0

- The port's link shall reject all U2 transition requests by the link partner.
- The PM timers shall be reset when this state is entered and is active.
- The port's link shall accept U1 entry requests by its link partner unless the hub has one or more packets/link commands to transmit on the port.
- If the U1 timeout is 0xFF, the port shall be disabled from initiating U1 entry but shall accept U1 entry requests by the link partner unless the hub has one or more packets/link commands to transmit on the port.
- If the U1 timeout is not 0xFF and the U1 timer reaches X, the port's link shall initiate a transition to U1.

U1_TIMEOUT = 0, U2_TIMEOUT = Y > 0

- The port's link shall reject all U1 transition requests by the link partner.
- The PM timers shall be reset when this state is entered and is active.
- The port's link shall accept U2 entry requests by its link partner unless the hub has one or more packets/link commands to transmit on the port.
- If the U2 timeout is 0xFF, the port shall be disabled from initiating U2 entry but shall accept U2 entry requests by the link partner unless the hub has one or more packets/link commands to transmit on the port.
- If the U2 timeout is not 0xFF and the U2 timer reaches Y, the port's link shall initiate a direct transition from U0 to U2. In this case, PORT_U2_TIMEOUT represents an amount of inactive time in U0.

U1_TIMEOUT =X > 0, U2_TIMEOUT = Y > 0

- The PM timers are reset when this state is entered and is active.
- The port's link shall accept U1 or U2 entry requests by its link partner unless the hub has one or more packets/link commands to transmit on the port.
- If the U1 timeout is 0xFF, the port shall be disabled from initiating U1 entry but shall accept U1 entry requests by the link partner unless the hub has one or more packets/link commands to transmit on the port.
- If the U1 timeout is not 0xFF and the U1 timer reaches X, the port's link shall initiate a transition to U1.

- If the U2 timeout is 0xFF, the port shall be disabled from initiating U2 entry but shall accept U2 entry requests by the link partner unless the hub has one or more packets/link commands to transmit on the port.

A port transitions to one of the Enabled U0 states (depending on the U1 and U2 Timeout values) in any of the following situations:

- From any state if the hub receives a SetPortFeature(PORT_LINK_STATE) U0 request.
- From U1 if the link partner successfully initiates a transition to U0.
- From U2 if the link partner successfully initiates a transition to U0.
- From U1 if the hub successfully initiates a transition to U0 after receiving a packet routed to the port.
- From U2 if the hub successfully initiates a transition to U0 after receiving a packet routed to the port
- From an attempt to transition from the U0 to the U1 state if the downstream port's link partner rejects the transition attempt
- From an attempt to transition from the U0 to the U2 state if the downstream port's link partner rejects the transition attempt
- From U3 if the upstream port of the hub receives wakeup signaling and the hub downstream port being transitioned received wakeup signaling while it was in U3.
- From U3 if the downstream port's link partner initiated wake signaling and the upstream hub port's link is not in U3.

Note: Refer to Section 10.1.4 for details on cases where a downstream port's link partner initiates remote wakeup signaling.

10.4.2.2 Attempt U0 – U1 Transition

In this state, the port attempts to transition its link from the U0 state to the U1 state.

A port shall attempt to transition to the U1 state in any of the following situations:

- The U1 timer reaches the U1 timeout value.
- The hub receives a SetPortFeature(PORT_LINK_STATE) U1 request.
- The downstream port's link partner initiates a U0-U1 transition.

If the transition attempt fails, the port returns to the appropriate enabled U0 state. However, if this state was entered due to a SetPortFeature request, the port continues to attempt the U0-U1 transition on its link.

Note: that the SetPortFeature request is typically only used for U1 entry for test purposes.

10.4.2.3 Attempt U0 – U2 Transition

In this state, the port attempts to transition the link from the U0 state to the U2 state.

A port shall attempt to transition to the U2 state in any of the following situations:

- The U2 timer reaches the U2 timeout value.
- The hub receives a SetPortFeature(PORT_LINK_STATE) U2 request.
- The downstream port's link partner initiates a U0-U2 transition.

If the transition attempt fails, the port returns to the appropriate enabled U0 state. However, if this state was entered due to a SetPortFeature request, the port continues to attempt the U0-U2 transition.

Note: that the SetPortFeature request is typically only used for U2 entry for test purposes.

10.4.2.4 Link in U1

Whenever a downstream port enters U1 and all downstream ports are now in the U1 or a lower power state, the hub shall initiate a transition to U1 on the upstream port within tHubPort2PortExitLat if the upstream port is enabled for U1.

The U2 timer is reset to zero and started when the Link enters U1.

If the U2 timeout is not 0xFF and the U2 timer reaches Y, the port's link shall initiate a direct transition from U1 to U2. In this case, PORT_U2_TIMEOUT represents an amount of time in U1.

Whenever a downstream port or its link partner initiates a transition from U1 to one of the Enabled U0 states and the upstream port is not in U0, the hub shall initiate a transition to U0 on the upstream port within tHubPort2PortExitLat of when the transition was initiated on the downstream port. If the upstream port is in U0, it shall remain in U0 while the downstream port transitions to U0.

10.4.2.5 Link in U2

The following rules apply when a downstream port enters U2:

- If all downstream ports are now in the U2 or a lower power state, the hub shall initiate a transition to U2 on the upstream port within tHubPort2PortExitLat, if the upstream port is enabled for U2. If U2 is not enabled on the upstream port, but U1 is enabled, the hub shall initiate a transition to U1 with the same timing requirements.
- If all downstream ports are now in the U1 or lower power state, the hub shall initiate a transition to U1 on the upstream port within tHubPort2PortExitLat, if the upstream port is enabled for U1.

Whenever a downstream port or its link partner initiates a transition from U2 to one of the Enabled U0 states and the hub upstream port is not in U0:

- If the hub upstream port's link is in U2, the hub shall initiate a transition to U0 on the upstream port's link within tHubPort2PortExitLat of when the transition was initiated on the downstream port.
- If the hub upstream port's link is in U1, the hub upstream port shall initiate a transition to U0 within tHubPort2PortExitLat + U2DevExitLat-U1DevExitLat of when the transition was initiated on the downstream port.

10.4.2.6 Link in U3

The following rules apply when a downstream port enters U3:

- If all downstream ports are now in the U2 or U3, the hub shall initiate a transition to the lowest enabled power state above U3 on the upstream port within tHubPort2PortExitLat.
- If all downstream ports are now in the U1 or lower power state, the hub shall initiate a transition to U1 on the upstream port within tHubPort2PortExitLat, if the upstream port is enabled for U1.

Refer to Section 10.3.1.5 for a detailed description of the transition from Enabled – U0 Only to the U3 state.

Note: If the upstream port of the hub receives a packet that is routed to a downstream port that is in U3, the packet is silently dropped. The hub shall perform normal link level acknowledgement of the header packet in this case.

10.5 Hub Upstream Facing Port

The following sections provide a functional description of a state machine that exhibits correct behavior for a hub upstream facing port. These sections also apply to the upstream facing port on a device unless exceptions are specifically noted. An upstream port shall only attempt to connect to the SuperSpeed and USB 2.0 interfaces as described by the upstream port state machine in the following sections.

Figure 10-11 is an illustration of the upstream facing port state machine. Each of the states is described in Section 10.5.1. In Figure 10-11, some of the entry conditions into states are shown without origin. These conditions have multiple origin states and the individual transitions lines are not shown so that the diagram can be simplified. The description of the entered state indicates from which states the transition is applicable.

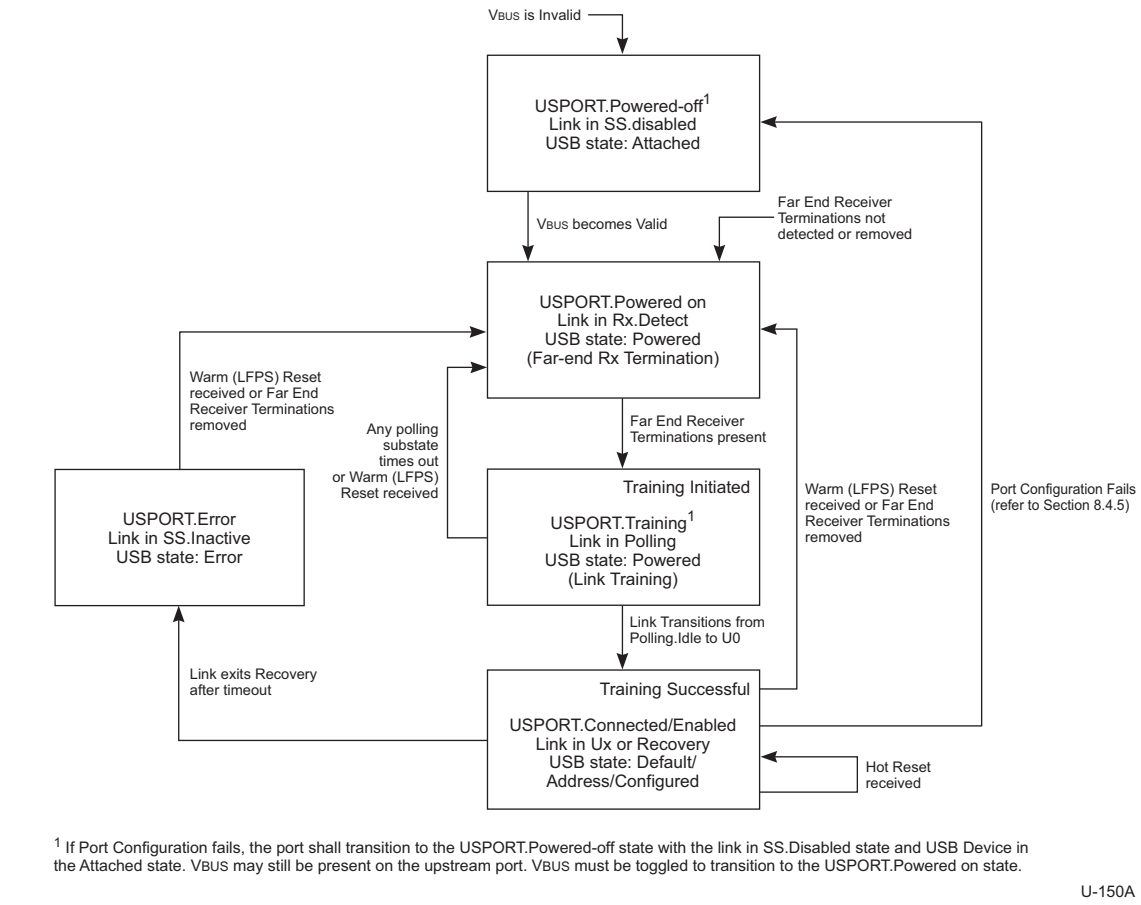


Figure 10-11. Upstream Facing Hub Port State Machine

10.5.1 Upstream Facing Port State Descriptions

Refer to Figure 9-1 for hub USB states.

10.5.1.1 USPORT.Powered-off

The USPORT.Powered-off state is the default state for an upstream facing port.

A port shall transition into this state if any of the following situations occur:

- From any state when VBUS is invalid.
- From any state if far-end receiver terminations are not detected.
- From the USPORT.Connected/Enabled state if the Port Configuration process fails.

In this state, the port's link shall be in the SS.Disabled state and the corresponding hub USB state shall be Attached.

Note: If the port enters this state because far end receiver terminations are not detected and VBUS is present, it may immediately transition to USPORT.Powered on without removing near end terminations.

10.5.1.2 USPORT.Powered-on

A port shall transition into this state in any of the following situations:

- From the USPORT.Powered-off state when VBUS becomes valid.
- From the USPORT.Error state when the link receives a warm reset or if Far-end Terminations are removed.
- From the USPORT.Connected/Enabled state when the link receives a Warm Reset.
- From the USPORT.Training state if the port's link times out from any Polling substate or if the port receives a Warm (LFPS) Reset.

In this state, the port's link shall be in the Rx.Detect state. The corresponding hub USB state shall be Powered (Far-end Receiver Termination substate). While in this state, if the USB 2.0 portion of the hub enters the suspended state, the total hub current draw from VBUS shall meet the suspend current limit.

10.5.1.3 USPORT.Training

A port transitions to this state from the USPORT.Powered-on state when SuperSpeed far-end receiver terminations are detected.

In this state, the port's link shall be in the Polling state. The corresponding hub USB state shall be Powered (Link Training substate).

10.5.1.4 USPORT.Connected/Enabled

A port transitions to this state from the USPORT.Training state when its link enters U0 from Polling.Idle. A port remains in this state during hot reset. When a hot reset is completed, the corresponding hub USB state shall transition to Default.

In this state, the port's link shall be in the U0, U1, U2, U3, or Recovery state. The corresponding hub USB state shall be Default, Address, or Configured.

When the link enters U0 the port shall start the port configuration process as defined in Section 8.4.5.

The port may send link management packets or link commands but shall not transmit any other packets except to respond to default control endpoint requests while in the USPORT.Connected state.

10.5.1.5 USPORT.Error

A port transitions to this state when a serious error condition occurred while attempting to operate the link. A port transitions to this state in any of the following situations:

- From the USPORT.Connected/Enabled state if the link enters Recovery and times out without recovering.

In this state, the port's link shall be in the SS.Inactive state. The corresponding hub USB state shall be Error.

A port exits the Error state only if a Warm Reset is received on the link or if Far-end Receiver Terminations are removed.

10.5.2 Hub Connect State Machine

The following sections provide a functional description of a state machine that exhibits correct hub behavior for when to connect on SuperSpeed or USB 2.0. For a hub the connection logic for SuperSpeed and USB 2.0 are completely independent. The hub shall follow the USB 2.0 specification for connecting on USB 2.0. Figure 10-12 is an illustration of the hub connect state machine for SuperSpeed. Each of the states is described in Section 10.5.2.1.

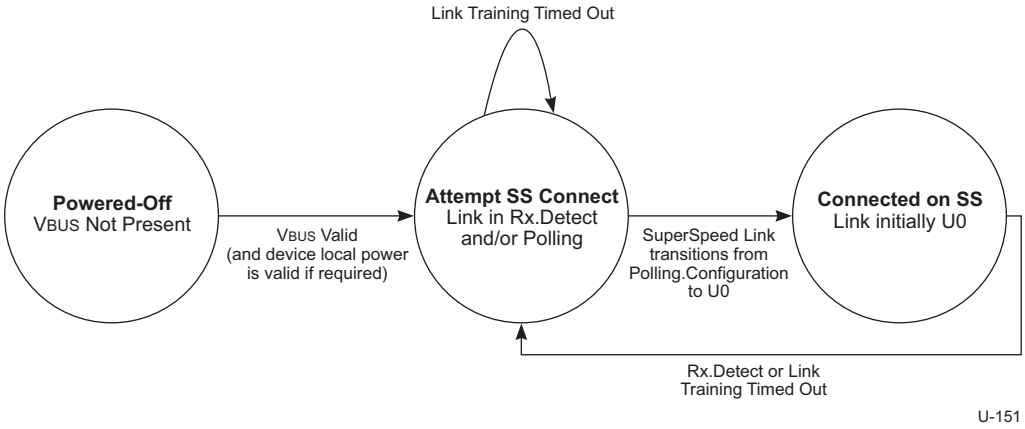


Figure 10-12. Hub Connect State Machine

10.5.2.1 Hub Connect State Descriptions

10.5.2.2 HCONNECT.Powered-off

The HCONNECT.Powered-off state is the default state for a hub device. A hub device shall transition into this state if the following situation occurs:

- From any state when VBUS is removed.

In this state, the hub upstream port's link shall be in the SS.Disabled state.

10.5.2.3 HCONNECT.Attempt SS Connect

A hub shall transition into this state if any of the following situations occur:

- From the HCONNECT.Powered-off state when VBUS becomes valid (and local power is valid if required).
- From the HCONNECT.Connected on SS state if Rx.Detect or Link Training time out.

In this state, the hub's upstream port SuperSpeed link is in Rx.Detect or Polling.

10.5.2.4 HCONNECT.Connected on SS

A port shall transition into this state if the following situation occurs:

- From the PCONNECT.Attempt SS Connect when the link transitions from polling to U0.

In this state the hub's upstream port SuperSpeed link is in U0, U1, U2, U3, Inactive, Rx.Detect, Recovery, or Polling.

10.6 Upstream Facing Port Power Management

The following sections provide a functional description of a state machine that exhibits correct link power management behavior for a hub upstream facing port.

Figure 10-13 is an illustration of the upstream facing port power management state machine. Each of the states is described in Section 10.4.2. In Figure 10-13, some of the entry conditions into states are shown without origin. These conditions have multiple origin states and the individual transitions lines are not shown so that the diagram can be simplified. The description of the entered state indicates from which states the transition is applicable.

If there is a status change on any downstream port, the hub shall initiate a transition on the upstream port's link to U0 if the upstream port is in U1 or U2.

If there is a status change on any downstream port and the hub upstream port's link is in U3, the hub behavior is specified by the current remote wakeup mask settings. Refer to Section 10.14.2.10 for more details.

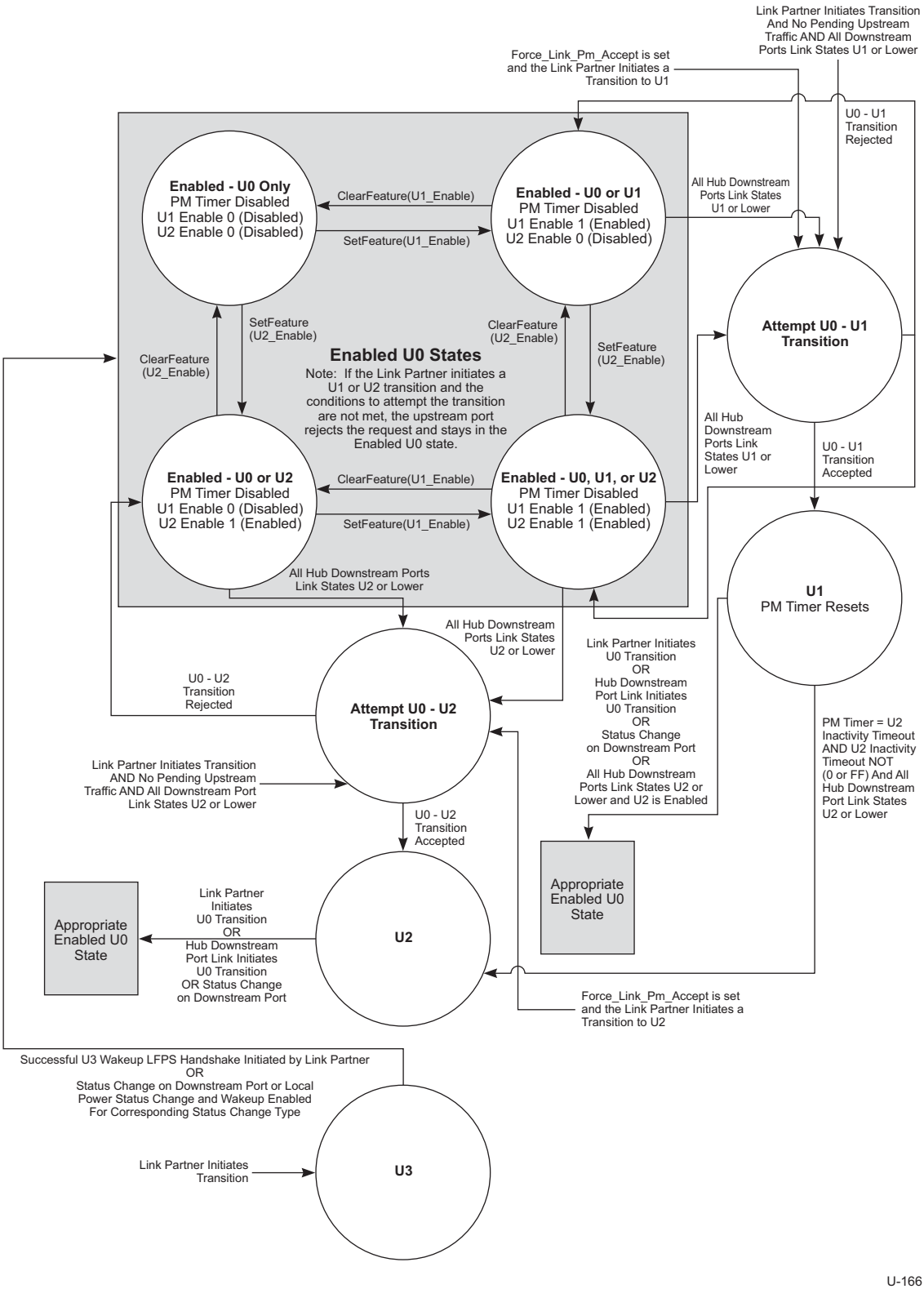


Figure 10-13. Upstream Facing Hub Port Power Management State Machine

10.6.1 Upstream Facing Port PM Timer

The hub upstream port maintains a logical PM timer for keeping track of when the U2 inactivity timeout is exceeded. No standard U1 inactivity timeout is defined. The U2 inactivity timeout is set when a U2 Inactivity Timeout LMP is received. The PM timer is reset when the hub upstream port link enters U1. The PM timer shall be accurate to +500/-0 µs. Other requirements for the timer are defined in the upstream port PM state machine descriptions.

10.6.2 Hub Upstream Facing Port State Descriptions

10.6.2.1 Enabled U0 States

There are four enabled U0 states that differ only in the U1 and U2 Enable settings. The following rules apply globally to all Enabled U0 states:

- The upstream port shall not initiate a transition to U1 or U2 if there are pending packets to transmit on the upstream port.
- The upstream port shall accept U1 or U2 transitions from the link partner if the Force_LinkPM_Accept bit is set to one (refer to Section 8.4.2).

The port behaves as follows for the various combinations of U1 and U2 Enable values:

U1_ENABLE = 0, U2_ENABLE = 0

- This is the default state before the hub has received any SetFeature(U1/U2_ENABLE) requests.
- The PM timer may be disabled and the PM timer values shall be ignored.
- The port's link shall accept U1 entry requests by its link partner unless the hub has one or more packets/link commands to transmit on the port or one or more of the hub downstream ports has a link in U0 or recovery.
- The port's link shall accept U2 entry requests by its link partner unless the hub has one or more packets/link commands to transmit on the port or one ore more of the hub downstream ports has a link in U0, U1, or recovery.
- The port's link shall not attempt to initiate transitions to U1 or U2.

U1_ENABLE = 1, U2_ENABLE = 0

- The port's link shall not initiate a U2 transition.
- The port's link shall accept all U2 entry requests by the link partner unless the hub has one or more packets/link commands to transmit on the port or one or more of the hub downstream ports has a link in U0, U1 or recovery.
- The port's link shall accept U1 entry requests by its link partner unless the hub has one or more packets/link commands to transmit on the port or one or more of the hub downstream ports has a link in U0 or recovery.
- The PM timer may be disabled and the PM timer values shall be ignored.
- The port's link shall initiate a transition to U1 if all the hub downstream ports are in U1 or a lower link state.

U1_ENABLE = 0, U2_ENABLE = 1

- The port's link shall not initiate a U1 transition.

- The port's link shall accept all U1 entry requests by the link partner unless the hub has one or more packets/link commands to transmit on the port or one or more of the hub downstream ports has a link in U0 or recovery.
- The port's link shall accept U2 entry requests by its link partner unless the hub has one or more packets/link commands to transmit on the port or one ore more of the hub downstream ports has a link in U0, U1, or recovery.
- The PM timer may be disabled and the PM timer values shall be ignored.
- The port's link shall initiate a transition to U2 if all the hub downstream ports are in U2 or a lower link state.

U1_ENABLE = 1, U2_ENABLE = 1

- The port's link shall accept U1 or U2 entry requests by its link partner unless the hub has one or more packets/link commands to transmit on the port.
  - A U1 entry request shall not be accepted if one or more of the hub downstream ports has a link in U0 or recovery.
  - A U2 entry request shall not be accepted if one or more of the hub downstream ports has a link in U0, U1, or recovery.
- The port's link shall initiate a transition to U1 if all the hub downstream ports are in U1 or a lower link state unless the conditions for U2 entry are satisfied.
- The port's link shall initiate a transition to U2 if all the hub downstream ports are in U2 or a lower link state. Note that if the port is already in U1, then the port shall transition to U0 before transitioning to U2.
- The PM timer may be disabled and the PM timer values shall be ignored.

A port transitions to one of the Enabled U0 states (depending on the U1 and U2 Enable values) in any of the following situations:

- From U1 if the link partner successfully initiates a transition to U0.
- From U2 if the link partner successfully initiates a transition to U0.
- From U1 if there is a status change on a downstream port.
- From U2 if there is a status change on a downstream port.
- From U1 if a hub downstream port's link initiates a transition to U0.
- From U2 if a hub downstream port's link initiates a transition to U0.
- From an attempt to transition from the U0 to the U1 state if the upstream port's link partner rejects the transition attempt
- From an attempt to transition from the U0 to the U2 state if the upstream port's link partner rejects the transition attempt
- From U3 if the upstream port of the hub receives wakeup signaling.
- From U3 if there is a status change on a downstream port or a local power status change and remote wakeup is enabled for the corresponding event type.

10.6.2.2 Attempt U0 – U1 Transition

In this state the port attempts to transition its link from the U0 state to the U1 state.

A port shall attempt to transition to the U1 state in any of the following situations:

- U1 entry is requested by the link partner and there is no pending traffic on the port and all the hub downstream port's links are in U1 or a lower state.

- All the hub downstream ports are in U1 or a lower link state and there is no pending traffic to transmit on the upstream port and U1_ENABLE is set to one.
- U1 entry is requested by the link partner and Force_LinkPM_Accept bit is set.

If the transition attempt fails (an LXU is received or the link goes to recovery), the port returns to the appropriate enabled U0 state.

10.6.2.3 Attempt U0 – U2 Transition

In this state, the port attempts to transition the link from the U0 state to the U2 state.

A port shall attempt to transition to the U2 state in any of the following situations:

- U2 entry is requested by the link partner and there is no pending traffic on the port and all the hub downstream port's links are in U2 or a lower state.
- All the hub downstream ports are in U2 or a lower link state and there is no pending traffic to transmit on the upstream port and U2_ENABLE is set to one.
- U2 entry is requested by the link partner and Force_LinkPM_Accept bit is set.

If the transition attempt fails (an LXU is received or the link goes to recovery), the port returns to the appropriate enabled U0 state.

10.6.2.4 Link in U1

The PM timer is reset when this state is entered and is active.

A port transitions to U1:

- After sending an LAU to accept a transition initiated by the link partner.
- After receiving an LAU from the link partner after initiating an attempt to transition the link to U1

If the U2 inactivity timeout is not 0xFF or 0x00, and the PM timer reaches the U2 inactivity timeout, the port's link shall initiate a transition from U1 to U2.

10.6.2.5 Link in U2

The link is in U2.

A port transitions to U2:

- After sending an LAU to accept a transition initiated by the link partner.
- After receiving an LAU from the link partner after initiating an attempt to transition the link to U2

10.6.2.6 Link in U3

The link is in U3.

A port transitions to U3:

- After sending an LAU to accept a transition initiated by the link partner.

10.7 Hub Header Packet Forwarding and Data Repeater

The Hub uses a store and forward model for header packets and a repeater model for data that combined provide the following general functionality.

In the downstream direction:

- Validates header packets
- Sets up connection to selected downstream port
- Forwards header packets to downstream ports
- Forwards data payload to downstream port if present
- Sets up and tears down connectivity on packet boundaries

In the upstream direction:

- Validates header packets
- Sets up connection to upstream port
- Forwards header packets to the upstream port
- Forwards data packet payload to upstream port if present
- Sets up and tears down connectivity on packet boundaries

10.7.1 Hub Elasticity Buffer

There are no direct specifications for elasticity buffer behavior in a hub. However, note that a hub must meet the requirements in Section 10.7.3 for the maximum variation in propagation delay for header packets that are forwarded from the hub upstream port to a downstream port.

10.7.2 SKP Ordered Sets

A hub transmits SKP ordered sets, following the rules for all transmitters in Chapter 7, for all transmissions.

10.7.3 Interpacket Spacing

When a hub originates or forwards packets, Data packet headers and data packet payloads shall be sent contiguously at all times as required in Section 7.2.1.

When a hub forwards a header packet downstream and the downstream port link is in U0 when the header packet is received on the hub upstream port the propagation delay variation shall not be more than tPropagationDelayJitterLimit.

10.7.4 Header Packet Buffer Architecture

The specification does not require a specific architecture for the header packet buffers in a hub. An example architecture that meets the functional requirements of this specification is shown in Figure 10-14 and Figure 10-15 to illustrate the functional behavior of a hub. Figure 10-14 shows a hub with a four header packet Rx buffer for the upstream port and a four header packet Tx buffer for each of the downstream ports. Figure 10-15 shows a four header packet Rx buffer for each of the downstream ports and a four header packet Tx buffer for the upstream port. The buffers shown in Figure 10-14 and Figure 10-15 are independent physical buffers.

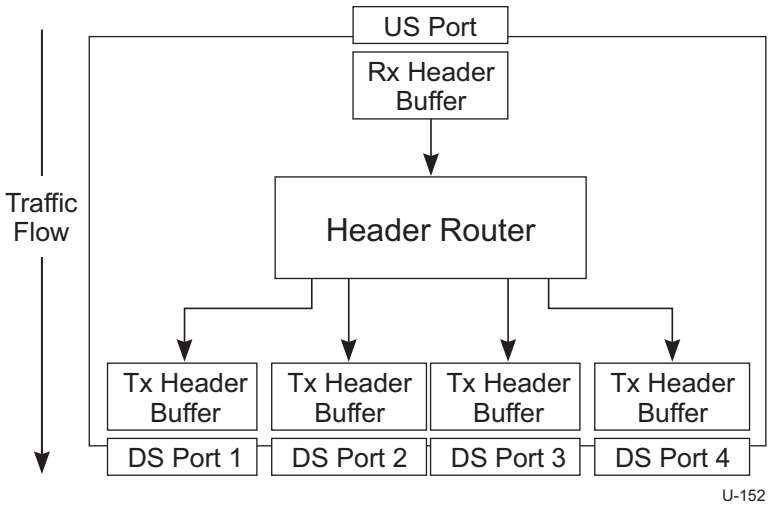


Figure 10-14. Example Hub Header Packet Buffer Architecture - Downstream Traffic

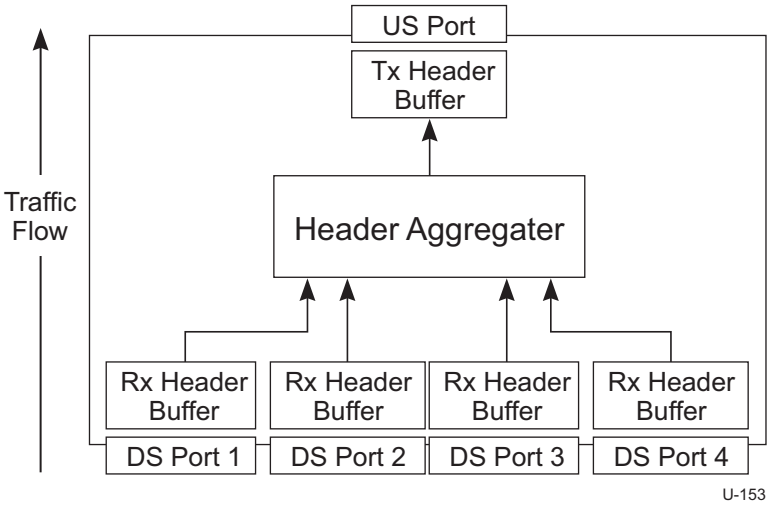


Figure 10-15. Example Hub Header Packet Buffer Architecture - Upstream Traffic

The following lists functional requirements for a hub buffer architecture with the assumption in each case that only the indicated port on the hub is receiving or transmitting header packets:

- A hub starting with all header packet buffers empty shall be able to receive at least eight header packets directed to the same downstream port that is not in U0 before its upstream port runs out of header packet flow control credits.

- A hub that receives a header packet on its upstream port that is routed to a downstream port shall immediately route the header packet to the appropriate downstream port header packet buffer (if space in that buffer is available) regardless of the state of any other downstream port header packet buffers or the state of the upstream port Rx header packet buffer. For example, a hub Tx header packet buffer for downstream port 1 is full and the hub has three more header packets routed to downstream port 1 in the hub upstream port Rx header packet buffer. If the hub now receives a header packet routed to downstream port 2, it must immediately route the header packet to the downstream port 2 Tx header packet buffer.
- A hub starting with all header packet buffers empty shall be able to receive at least eight header packets on the same downstream port directed for upstream transmission when the upstream port is not in U0.
- Header packets transmitted by a downstream port shall be transmitted in the order they were received on the upstream port.
- Header packets transmitted by an upstream port from the same downstream port shall be transmitted in the order they were received on that downstream port.

Sections 10.7.6, 10.7.8, 10.7.10, and 10.7.12 provide detailed functional state machines for the upstream and downstream port Tx and Rx header packet buffers in a hub implementation.

10.7.5 Upstream Facing Port Tx

This section describes the functional requirements of the upstream facing port Tx state machine.

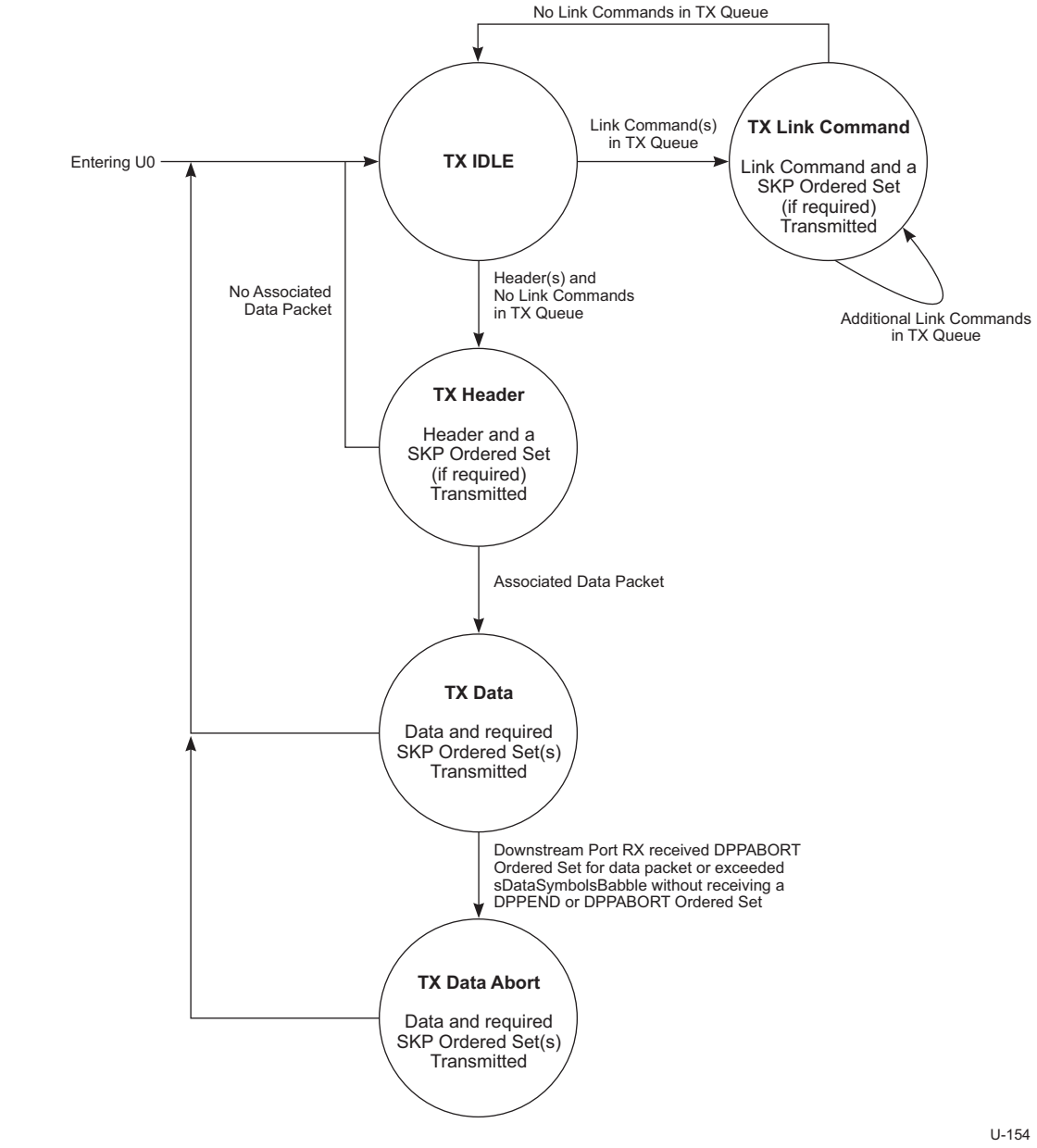


Figure 10-16. Upstream Facing Port Tx State Machine

10.7.6 Upstream Facing Port Tx State Descriptions

An upstream port shall maintain a count of transmitted symbols.

10.7.6.1 Tx IDLE

In the Tx IDLE state, the upstream port transmitter is actively transmitting idle symbols. A port shall transition to the Tx IDLE state in any of the following situations:

- From the Tx Data, Tx Data Abort, or Tx Header state after any required SKP ordered sets are transmitted.
- From the Tx Link Command state after a link command is transmitted and there are no other link commands awaiting transmission.
- As the default state when the link enters U0.

The transmitter shall transmit a LUP when required as described in Chapter 7.

When the transmitted symbol count reaches nSkipSymbolLimit, a SKP ordered set shall be transmitted and the transmitted symbol count shall be reset to zero.

10.7.6.2 Tx Header

In the Tx Header state, the upstream port transmitter is actively transmitting a header packet.

Note: A hub shall not abort the transmission of a header packet with a DPPABORT ordered set.

A port shall transition to the Tx Header state in any of the following situations:

- From the Tx IDLE state when there are one or more header packets queued for transmission and there are no link commands queued for transmission.

When the transmitted symbol count is greater than or equal to nSkipSymbolLimit at the end of transmitting any header packet except a data packet header packet with a data packet payload, a SKP ordered set shall be transmitted and the transmitted symbol count shall be reduced by nSkipSymbolLimit.

10.7.6.3 Tx Data

In the Tx Data state, the upstream port transmitter is actively transmitting a data packet payload. After transmitting the end framing symbols for the data packet payload and required SKP ordered sets, the port may remove the data packet payload from hub storage. A hub shall not retransmit a DPP packet under any circumstances.

A port shall transition to the Tx Data state from the Tx Header state when there is a data packet payload associated with the data packet header that was transmitted. The data packet payload transmission shall begin immediately after transmission of the last symbol of the data packet header.

At the end of transmitting a data packet payload that is not aborted:

- While the transmitted symbol count is greater than or equal to nSkipSymbolLimit, a SKP ordered set shall be transmitted and the transmitted symbol count shall be reduced by nSkipSymbolLimit.
- The sequence is repeated until the symbol count is less than nSkipSymbolLimit.

10.7.6.4 Tx Data Abort

In the Tx Data abort state, the upstream port transmitter aborts the normal transmission of a data packet payload by transmitting DPPABORT ordered set and required SKP ordered sets. The port then removes the data packet payload from hub storage.

A port shall transition to the Tx Data Abort state from the Tx Data state when the downstream port receiving the data packet payload receives a DPPABORT ordered set or has received sDataSymbolsBabble symbols without receiving a valid DPPEND ordered set or DPPABORT ordered set.

At the end of transmitting a DPPABORT ordered set:

- While the transmitted symbol count is greater than or equal to nSkipSymbolLimit, a SKP ordered set shall be transmitted and the transmitted symbol count shall be reduced by nSkipSymbolLimit.
- The sequence is repeated until the symbol count is less than nSkipSymbolLimit.

10.7.6.5 Tx Link Command

In the Tx Link Command state, the upstream port transmitter is actively transmitting a link command. If multiple link commands are queued to be transmitted in the Tx Link Command state, they shall be transmitted without a gap unless SKP ordered sets are transmitted.

A port shall transition to the Tx Link Command state in any of the following situations:

- From the Tx IDLE state when there are one or more link commands queued for transmission.
- From the Tx Link Command state when there are additional link commands queued for transmission.

When the transmitted symbol count is greater than or equal to nSkipSymbolLimit at the end of transmitting any link command, a SKP ordered set shall be transmitted and the transmitted symbol count shall be reduced by nSkipSymbolLimit.

10.7.7 Upstream Facing Port Rx

This section describes the functional requirements of the upstream facing port Rx state machine.

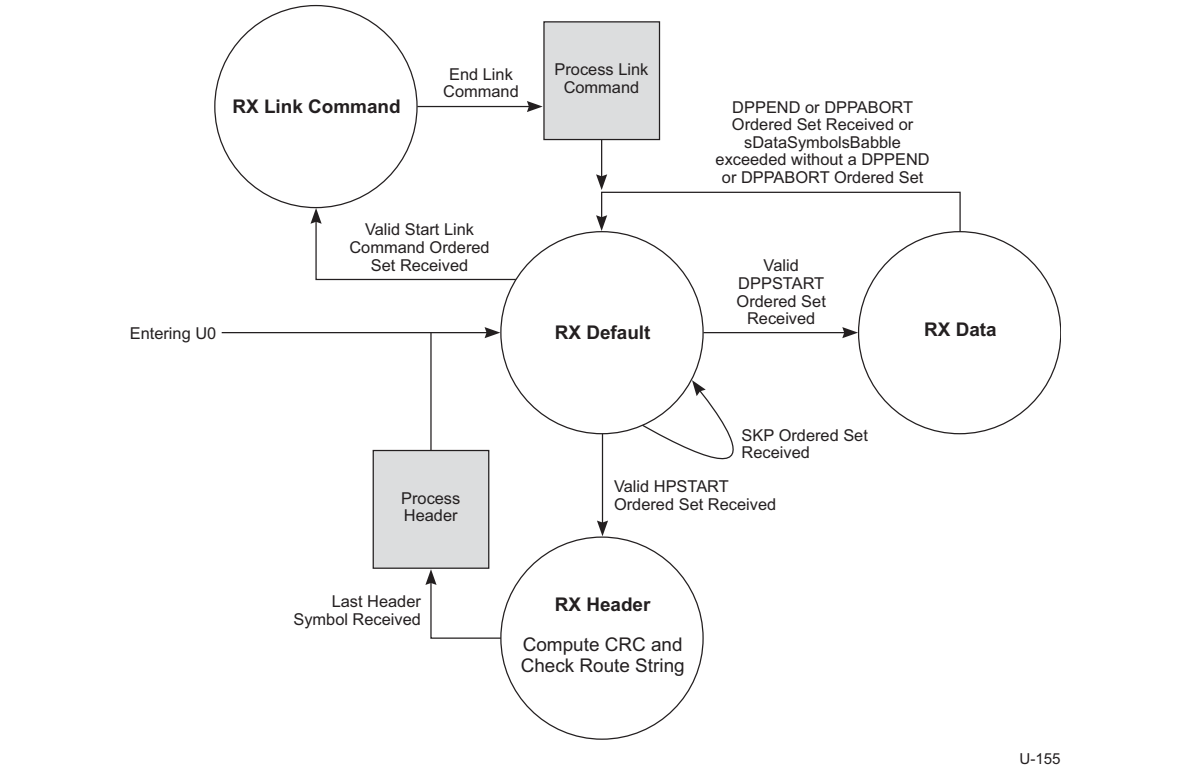


Figure 10-17. Upstream Facing Port Rx State Machine

10.7.8 Upstream Facing Port Rx State Descriptions

10.7.8.1 Rx Default

In the Rx Default state, the upstream port receiver is actively processing received symbols and looking for DPPSTART ordered set, HPSTART ordered set, or LCSTART ordered set framing symbols to begin receiving a packet or link command.

If a DPPStart ordered set is received that did not immediately follow a DPH, it is ignored and the port receiver stays in the RX.Default state.

A port shall transition to the Rx Default state in any of the following situations:

- From the Rx Data state when a DPPEND ordered set or DPPABORT ordered set is received.
- From the Rx Header state when the last symbol in a header packet is received.
- From the Rx Data state when sDataSymbolsBabble is reached without receiving a DPPEND ordered set or DPPABORT ordered set.
- After receiving a link command.
- As the default state when the link enters U0.

10.7.8.2 Rx Data

In the Rx Data state, the upstream port receiver is actively processing received symbols and looking for a DPPEND ordered set or DPPABORT ordered set. The receiver shall start a count of received symbols from zero when entering this state. The first symbol counted is the first symbol after the DPPSTART ordered set.

A port shall transition to the Rx Data state when it receives a valid DPPSTART ordered set.

When the port detects an error before the end of the DPP as defined in Section 7.2.4.1.6, it may clear the data from its buffers only after it has transmitted the received DPP including the DPPABORT ordered set on the appropriate downstream port. The hub shall transmit the valid received symbols before the error for a data packet payload followed by a DPPABORT ordered set on the appropriate downstream port when an error is detected. Note that this requirement applies even when the hub detects an error before beginning to transmit the DPP on the hub downstream port.

The hub shall have at least 1080 bytes of buffering for data packets received on the upstream port.

10.7.8.3 Rx Header

In the Rx header state, the upstream port receiver is actively processing received symbols until the last header packet symbol is received. The receiver shall start a count of received symbols at zero when entering this state. The first symbol counted is the first symbol after the HPSTART ordered set.

A port shall transition to the Rx Header state when it receives a valid HPSTART ordered set.

The port shall finish validating CRC-16, the Link Control Word CRC-5, and check the route string and header packet type within four symbol times after the last symbol of the header packet is received.

Implementations may have to begin the CRC calculation as symbols are received and check the route string before the header packet is verified to meet this requirement.

10.7.8.4 Process Header Packet

When the final symbol for a header packet is received the port shall perform all additional processing necessary for the header packet. Any such processing shall not block the port from immediately returning to the Rx Default state and continuing to process received symbols.

A port performs additional header packet processing in any of the following situations. The additional processing steps in each situation are described.

- When the final header packet symbol is received in the Rx Header state and the header packet CRC-16 and Link Control Word CRC-5 are determined to be valid, the appropriate LGOOD_n link command is queued for transmission by the receiving port. Then if the upstream port Rx header packet buffer has at least four free slots, the appropriate LCRD_x link command shall be queued for transmission by the upstream port. Otherwise, the appropriate LCRD_x link command shall be queued for transmission once the Rx header packet buffer slot used for the header packet is available.
  Note: A hub implementation could choose to provide more than four storage slots in an Rx header packet buffer.

— If the header packet is routed to a downstream port that is not in U0 (and the header packet is not an ITP):
  1. If the appropriate downstream port link is in U1 or U2 and not in Recovery, the hub initiates U0 entry on the appropriate downstream port link. U0 entry shall be initiated tDownLinkStateChange from when the hub received the route string of the header packet.
  2. The header packet is marked deferred (if it is not already marked deferred) and the correct Link Control Word CRC-5 is re-calculated for modified header packet.
  3. If the header packet was marked deferred in step 2, a deferred header packet including the hub depth and correct CRC-5 is queued into the upstream port Tx header packet buffer.
  4. The header packet is queued for transmission on the appropriate downstream port.
  5. If the header packet is a data packet header the corresponding data packet payload is discarded.

— If the header packet is routed to a downstream port that is in U0 or the header packet is an ITP (in which case the processing below is done independently for each downstream port with a link in U0):
  1. If the downstream port Tx header packet buffer queue is not empty (there is at least one header packet in the queue that has not been completely transmitted) or no link credit is available for transmission on the downstream port, the header packet is marked delayed and the correct Link Control Word CRC-5 is re-calculated for modified header packet.
  2. The header packet is queued for transmission on the appropriate downstream port.

  Note: If the queue for the appropriate downstream port is full, the header packet is queued as soon as a space is available in the appropriate downstream port queue. The hub shall still process subsequent header packets normally while a downstream port queue is full if they are directed to a different downstream port.

— If the header packet is an ITP:
  1. The header packet is queued for transmission on each downstream port with a link in U0
  2. If the downstream port Tx header packet buffer queue is not empty (there is at least one header packet in the queue that has not been completely transmitted) or no link credit is available for transmission on the downstream port or the delay introduced will exceed tPropagationDelayJitterLimit, the header packet is marked delayed and the correct Link Control Word CRC-5 is re-calculated for modified header packet.

  Note: If the queue for the appropriate downstream port is full, the header packet is queued as soon as a space is available in the appropriate downstream port queue. The hub shall still process subsequent header packets normally while a downstream port queue is full if they are directed to a different downstream port.

— If the header packet is not routed to a downstream port:
  1. The header packet is processed.
  2. The header packet is removed from the RX header packet buffer.
  3. A response to the header packet is queued for transmission if required.

— If the header packet is routed to a disabled or nonexistent downstream port:

1. The header packet is removed from the RX header packet buffer.
2. The header packet is silently discarded.

- When the final header packet symbol is received in the Rx header packet state and either the header packet CRC-16 or Link Control Word CRC-5 is determined to be invalid.
  — The header packet is removed from the Rx header packet buffer.
  — An LBAD link command is queued for transmission by the upstream port.

  Note: All subsequent header packets are silently discarded until a retry of the invalid header packet is received.

10.7.8.5 Rx Link Command

In the Rx Link Command state, the upstream port receiver is actively processing received symbols and looking for the end of a link command (second instance of link command word).

A port transitions to the Rx Link Command state when it receives a valid LCSTART ordered set.

10.7.8.6 Process Link Command

Once the final symbol for a link command is received, the port shall perform all additional processing necessary for the link command. Any such processing shall not block the port from immediately returning to the Rx Default state and continuing to process received symbols.

10.7.9 Downstream Facing Port Tx

This section describes the functional requirements of the downstream facing port Tx state machine.

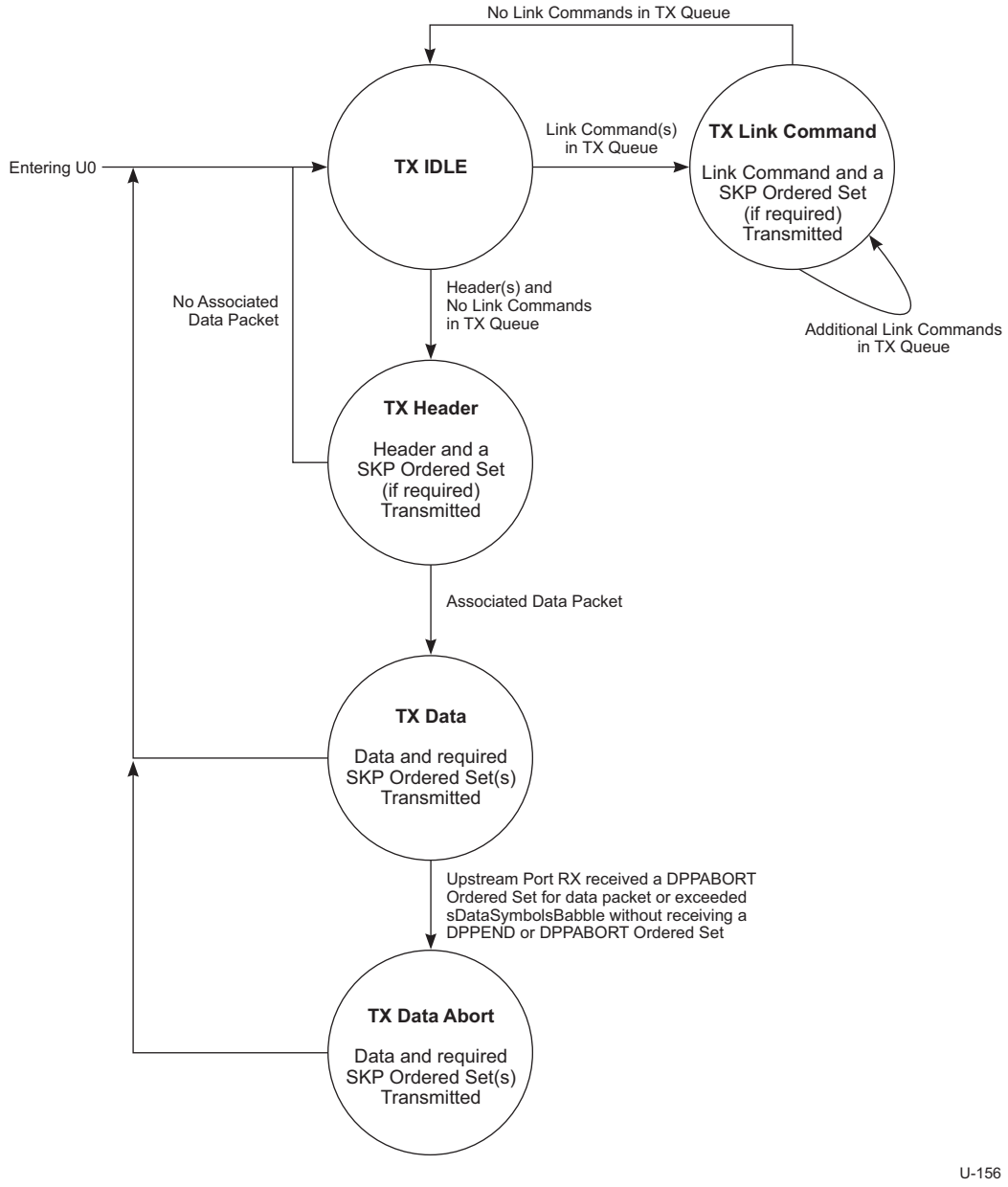


Figure 10-18. Downstream Facing Port Tx State Machine

10.7.10 Downstream Facing Port Tx State Descriptions

A downstream port shall maintain a count of transmitted symbols.

10.7.10.1 Tx IDLE

In the Tx IDLE state, the downstream port transmitter is actively transmitting idle symbols. A port shall transition to the TX IDLE state in any of the following situations:

- From the Tx Data, Tx Data Abort, or Tx Header state after the last required SKP ordered set is transmitted.
- From the Tx Link Command state after a link command is transmitted and there are no other link commands awaiting transmission.
- As the default state when the link enters U0.

When the transmitted symbol count reaches nSkipSymbolLimit a SKP ordered set shall be transmitted and the transmitted symbol count shall be reset to zero.

10.7.10.2 Tx Header

In the Tx Header state, the downstream port transmitter is actively transmitting a header packet.

Note: A hub shall not abort the transmission of a header packet with a DPPABORT ordered set.

A port shall transition to the Tx Header state in any of the following situations:

- From the Tx IDLE state when there are one or more header packets queued for transmission and there are no link commands queued for transmission.

When the transmitted symbol count is greater than or equal to nSkipSymbolLimit at the end of transmitting any header packet except a data packet header packet with a data packet payload, a SKP ordered set shall be transmitted and the transmitted symbol count shall be reduced by nSkipSymbolLimit.

10.7.10.3 Tx Data

In the Tx Data state, the downstream port transmitter is actively transmitting a data packet payload. After transmitting the end framing symbols for the data packet payload and required SKP ordered sets the port may remove the data packet payload from hub storage. A hub shall not retransmit a data packet payload packet under any circumstances.

A port shall transition to the Tx Data state from the Tx Header state when there is a data packet payload associated with the data packet header packet that was transmitted. The data packet payload transmission shall begin immediately after transmission of the last symbol of the data packet header packet.

At the end of transmitting a data packet payload that is not aborted:

- While the transmitted symbol count is greater than or equal to nSkipSymbolLimit, a SKP ordered set shall be transmitted and the transmitted symbol count shall be reduced by nSkipSymbolLimit.
- The sequence is repeated until the symbol count is less than nSkipSymbolLimit.

10.7.10.4 Tx Data Abort

In the Tx Data abort state, the downstream port transmitter aborts the normal transmission of a data packet payload by transmitting DPPABORT ordered set framing symbols and required SKP ordered sets. The port then removes the data packet payload from hub storage.

A port transitions to the Tx Data Abort state from the Tx Data state when the upstream port receiving the data packet payload receives a DPPABORT ordered set or has received sDataSymbolsBabble symbols without receiving a valid DPPEND ordered set or DPPABORT ordered set.

At the end of transmitting a DPPABORT ordered set:

- While the transmitted symbol count is greater than or equal to nSkipSymbolLimit, a SKP ordered set shall be transmitted and the transmitted symbol count shall be reduced by nSkipSymbolLimit.
- The sequence is repeated until the symbol count is less than nSkipSymbolLimit.

10.7.10.5 Tx Link Command

In the Tx Link Command state, the downstream port transmitter is actively transmitting a link command. If multiple link commands are queued to be transmitted in the Tx Link Command state, they shall be transmitted without a gap unless SKP ordered sets are transmitted.

A port shall transition to the Tx Link Command state in any of the following situations:

- From the Tx IDLE state when there are one or more link commands queued for transmission.
- From the Tx Link Command state when there are additional link commands queued for transmission.

When the transmitted symbol count is greater than or equal to nSkipSymbolLimit at the end of transmitting any link command, a SKP ordered set shall be transmitted and the transmitted symbol count shall be reduced by nSkipSymbolLimit.

10.7.11 Downstream Facing Port Rx

This section describes the functional requirements of the downstream facing port Rx state machine.

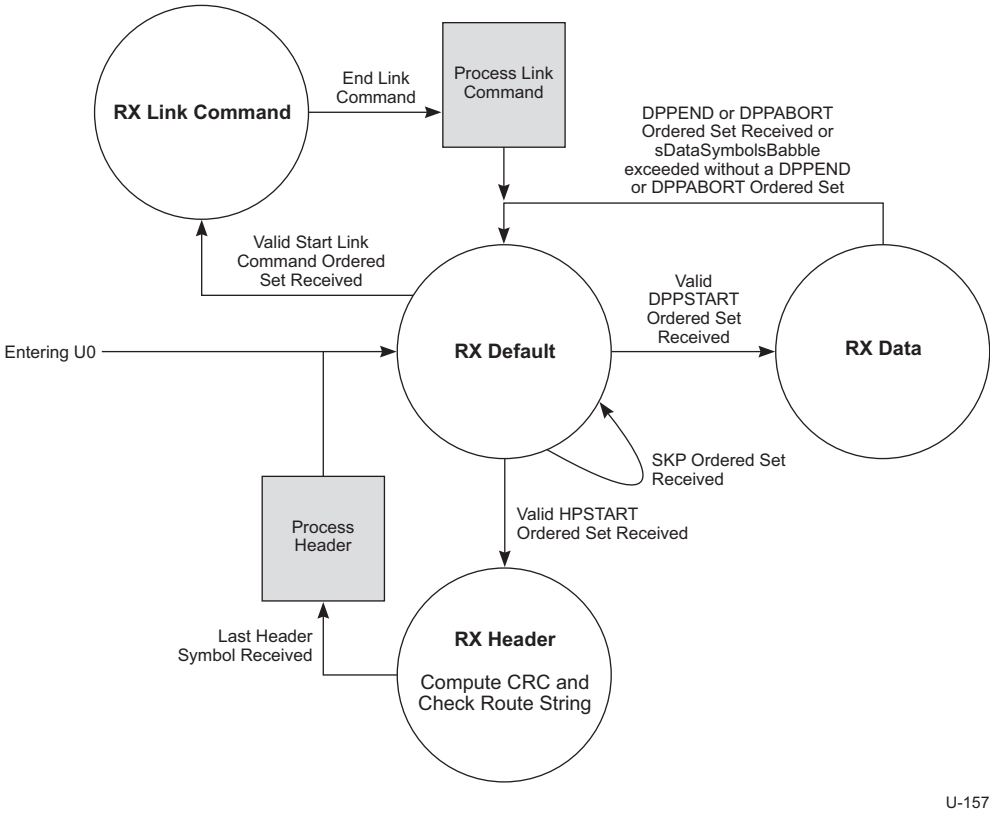


Figure 10-19. Downstream Facing Port Rx State Machine

10.7.12 Downstream Facing Port Rx State Descriptions

10.7.12.1 Rx Default

In the Rx Default state, the downstream port receiver is actively processing received symbols and looking for DPPSTART ordered set, HPSTART ordered set, or LCSTART ordered set framing symbols to begin receiving a packet or link command.

If a DPPStart ordered set is received that did not immediately follow a DPH, it is ignored and the port receiver stays in the RX.Default state.

A port shall transition to the Rx IDLE state in any of the following situations:

- From the Rx Data state when a DPPEND ordered set or DPPABORT ordered set is received.
- From the Rx Header state when the last symbol of a header packet is received.
- From the Rx Data state when sDataSymbolsBabble is reached without receiving a DPPEND ordered set or DPPSTART ordered set.
- After receiving a link command.
- As the default state when the link enters U0.

10.7.12.2 Rx Data

In the Rx Data state the downstream port receiver is actively processing received symbols and looking for a DPPEND ordered set, or DPPSTART ordered set. The receiver shall start a count of received symbols at zero when entering this state. The first symbol counted is the first symbol after the DPPSTART ordered set.

A port shall transition to the Rx Data state when it receives a valid DPPSTART ordered set.

When a downstream port detects an error before the end of the DPP as defined in Section 7.2.4.1.6 port it may clear the data from its buffers only after it has transmitted the received DPP including the DPPABORT ordered set on the upstream port. The hub shall transmit the valid received symbols before the error for a data packet payload followed by a DPPABORT ordered set on its upstream port when an error is detected on the downstream port. Note that this requirement applies even when the hub detects an error before beginning to transmit the data payload packet on the hub upstream port.

The hub shall have at least 1080 bytes of shared buffering for data packets received on all downstream ports.

10.7.12.3 Rx Header

In the Rx header state, the downstream port receiver is actively processing received symbols until the last header packet symbol is received. The receiver shall start a count of received symbols at zero when entering this state. The first symbol counted is the first symbol after the HPSTART ordered set.

A port shall transition to the Rx Header state when it receives a valid HPSTART ordered set.

The port shall finish validating the CRC-16 and the Link Control Word CRC-5 and check the header packet type within four symbol times after the last symbol of the header packet is received.

Note: Implementations may have to begin the CRC calculation as symbols are received to meet this requirement.

10.7.12.4 Process Header

When the final symbol for a header packet is received, the port shall perform all additional processing necessary for the header packet. Any such processing shall not block the port from immediately returning to the Rx Default state and continuing to process received symbols.

A port performs additional header packet processing in any of the following situations. The additional processing steps in each situation are described.

- When the final header packet symbol is received in the Rx header state and the header packet CRC-16 and Link Control Word CRC-5 are determined to be valid, the appropriate LGOOD_n link command is queued for transmission by the receiving port and:
  1. The header packet is queued for transmission on the upstream port.

     If the queue for the upstream port is full, the header packet is queued as soon as a space is available in the upstream port queue. The hub shall still process subsequent header packets normally while the upstream port queue is full. If header packets have been received on more than one downstream port or are queued to be sent by the hub controller when a space becomes available in the upstream port header packet queue, the hub shall prioritize a non-data packet header over a data packet header packet if one is waiting at the front of a downstream queue or from the hub controller. Otherwise, the arbitration algorithm the hub uses is not specified.
     Note: These arbitration requirements only apply across multiple downstream ports and the hub controller. For a single source (downstream port or hub controller), packets must be transmitted in the ordered received or generated.

  2. If the downstream port Rx header buffer has at least four free slots, the appropriate LCRD_x link command is queued for transmission by the downstream port. Otherwise, the appropriate LCRD_x link command is queued for transmission once the Rx header buffer slot used for the header packet is available.
- When the final header packet symbol is received in the Rx header state and either the header CRC-16 or Link Control Word CRC-5 is determined to be invalid:
  - The header packet is removed from the Rx header buffer.
  - An LBAD link command is queued for transmission by the downstream port.
  - Note: All subsequent header packets are silently discarded until a retry of the invalid header packet is received.

10.7.12.5 Rx Link Command

In the Rx Link Command state, the downstream port receiver is actively processing received symbols and looking for the end of a link command (second instance of link command word).

A port shall transition to the Rx Link Command state when it receives a valid LCSTART ordered set.

10.7.12.6 Process Link Command

Once the final symbol for a link command is received, the port shall perform all additional processing necessary for the link command. Any such processing shall not block the port from immediately returning to the Rx.Default state and continuing to process received symbols.

10.7.13 SuperSpeed Packet Connectivity

The SuperSpeed hub packet repeater/forwarder must reclock the packets in both directions. Reclocking means that the repeater extracts the data from the received stream and retransmits the stream using its own local clock.

10.8 Suspend and Resume

Hubs must support suspend and resume both as a USB device and in terms of propagating suspend and resume signaling. Global suspend/resume refers to the entire bus being suspended or resumed without affecting any hub's downstream facing port states; selective suspend/resume refers to a downstream facing port of a hub being suspended or resumed without affecting the hub state. SuperSpeed hubs only support selective suspend and resume. They do not support global suspend and resume. Selective suspend/resume is implemented via requests to a hub. Device-initiated resume is called remote-wakeup.

The hub follows the same suspend requirements as a SuperSpeed device on its upstream facing port.

When a hub downstream port link is in the U3 state, the following requirements apply to the hub if it receives wakeup signaling from its link partner on that downstream port:

- If the hub upstream port's link is not in U3, the hub shall drive remote wakeup signaling on the downstream link where the wakeup signaling was received in tHubDriveRemoteWakeDownstream.
- If the hub upstream port's link is in U3, the hub shall drive wakeup signaling on its upstream port in tHubPropRemoteWakeUpstream.
- If the hub upstream port is in the process of entering U3, the hub shall wait until the U3 entry is completed, before driving wakeup signaling on its upstream port in tHubPropRemoteWakeUpstream.

When a hub upstream port's link enters the U3 state and one of its downstream links is in U0/U1/U2/Recovery and has received a remote wake, the hub shall automatically drive remote wakeup on upstream port in tHubPropRemoteWakeUpstream.

When a hub upstream port's link is in the U3 state and it receives wakeup signaling from its link partner on the hub upstream port's link, the hub shall automatically drive remote wakeup to any downstream ports that are in U3 and have received remote wakeup signaling since entering U3.

If the hub upstream port's link is in U3, the hub shall drive wakeup signaling on its upstream port due to connect (when the downstream port enters DSPORT.Enabled), disconnect, or overcurrent events, if the hub is enabled for remote wakeup.

When the hub receives a SetPortFeature(PORT_LINK_STATE) U0 for a downstream port with a link in U3, the hub shall drive remote wakeup signaling on the link in tHubDriveRemoteWakeDownstream.

10.9 Hub Upstream Port Reset Behavior

Reset signaling to a hub is defined only in the downstream direction, which is at the hub's upstream facing port. The reset signaling mechanism required of the hub is described in Chapter 6.

A suspended hub shall interpret the start of reset as a wakeup event; it shall be awake and have completed its reset sequence by the end of reset signaling.

After completion of a Warm Reset, the entire hub returns to the default state.

After completion of a Hot Reset, the hub returns to the default state except port configuration information is maintained for the upstream port.

Irrespective of how the hub was reset, the hub needs to propagate reset as described in Section 10.3.1.6 and not just transition those downstream ports to the default state.

10.10 Hub Port Power Control

Self-powered hubs may have power switches that control delivery of power to downstream facing ports but it is not required. A hub with power switches can switch power to all ports as a group/gang, to each port individually, or have an arbitrary number of gangs of one or more ports.

A hub indicates whether or not it supports power switching by the setting of the Logical Power Switching Mode field in *wHubCharacteristics*. If a hub supports per-port power switching, then the power to a port is turned on or off as specified in Table 10-2. If a hub supports ganged power switching, then the power to all ports in a gang is turned on when power is required to be on for any port in the gang. The power to a gang is not turned off unless all ports in a gang are in a state that allows power to be removed as specified in Table 10-2. The power to a port is not turned on by a SetPortFeature(PORT_POWER) if both C_HUB_LOCAL_POWER and Local Power Status (in *wHubStatus*) are set to one at the time when the request is executed and the PORT_POWER feature would be turned on. A hub that supports charging applications may keep power on at other times. Refer to Section 10.3.1.1 for more details on allowed behavior for a hub that supports charging applications.

Although a self-powered hub is not required to implement power switching, the hub shall support the Powered-off states for all ports.

For a hub with no power switches, *bPwrOn2PwrGood* shall be set to zero.

10.10.1 Multiple Gangs

A hub may implement any number of power and/or over-current gangs. A hub that implements more than one over-current and/or power switching gang shall set both the Logical Power Switching Mode and the Over-current Reporting Mode to indicate that power switching and over-current reporting are on a per port basis (these fields are in *wHubCharacteristics*).

When an over-current condition occurs on an over-current protection device, the over-current is signaled on all ports that are protected by that device. When the over-current is signaled, all the ports in the group are placed in the DSPORT.Powered-off or the DSPORT.Powered-off-reset state, and the C_PORT_OVER_CURRENT field is set to one on all the ports. When port status is read from any port in the group, the PORT_OVER_CURRENT field will be set to one as long as the

over-current condition exists. The C_PORT_OVER_CURRENT field shall be cleared in each port individually.

When multiple ports share a power switch, setting PORT_POWER on any port in the group will cause the power to all ports in the group to turn on. It will not, however, cause the other ports in that group to leave the DSPORT.Powered-off or the DSPORT.Powered-off-reset state. When all the ports in a group are in the DSPORT.Powered-off state or the hub is not configured, the power to the ports is turned off.

If a hub implements both power switching and over-current, it is not necessary for the over-current groups to be the same as the power switching groups.

If an over-current condition occurs and power switches are present, then all power switches associated with an over-current protection circuit shall be turned off. If multiple over-current protection devices are associated with a single power switch, then that switch will be turned off when any of the over-current protection circuits indicates an over-current condition.

10.11 Hub Controller

The Hub Controller is logically organized as shown in Figure 10-20.

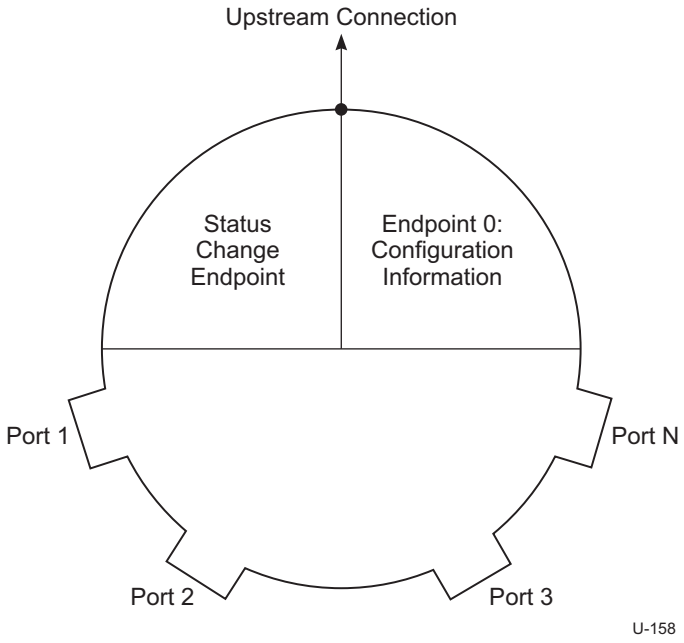


Figure 10-20. Example Hub Controller Organization

10.11.1 Endpoint Organization

The Hub Class defines one additional endpoint beyond the default control pipe, which is required for all hubs: the Status Change endpoint. This endpoint has the maximum burst size set to one. The host system receives port and hub status change notifications through the Status Change

endpoint. The Status Change endpoint is an interrupt endpoint. If no hub or port status change bits are set, then the hub returns an NRDY when the Status Change endpoint receives an IN (via an ACK TP) request. When a status change bit is set, the hub will send an ERDY TP to the host. The host will subsequently ask the Status Change endpoint for the data, which will indicate the entity (hub or port) with a change bit set. The USB system software can use this data to determine which status registers to access in order to determine the exact cause of the status change interrupt.

10.11.2 Hub Information Architecture and Operation

Figure 10-21 shows how status, status change, and control information relate to device states. Hub descriptors and Hub/Port Status and Control are accessible through the default control pipe. The Hub descriptors may be read at any time. When a hub detects a change on a port or when the hub changes its own state, the Status Change endpoint transfers data to the host in the form specified in Section 10.11.4.

Hub or port status change bits can be set because of hardware or software events. When set, these bits remain set until cleared directly by the USB system software through a ClearPortFeature() request or by a hub reset. While a change bit is set, the hub continues to report a status change when the Status Change endpoint is read until all change bits have been cleared by the USB system software.

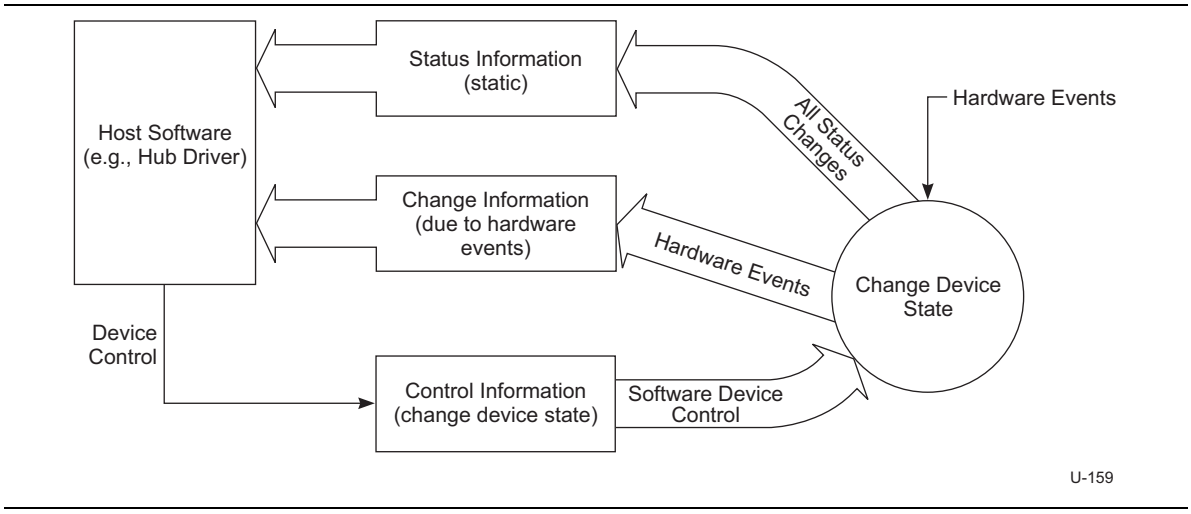


Figure 10-21. Relationship of Status, Status Change, and Control Information to Device States

The USB system software uses the interrupt pipe associated with the Status Change endpoint to detect changes in hub and port status.

10.11.3 Port Change Information Processing

Hubs report a port's status through port commands on a per-port basis. The USB system software acknowledges a port change by clearing the change state corresponding to the status change reported by the hub. The acknowledgment clears the change state for that port so future data transfers to the Status Change endpoint do not report the previous event. This allows the process to repeat for further changes (see Figure 10-22).

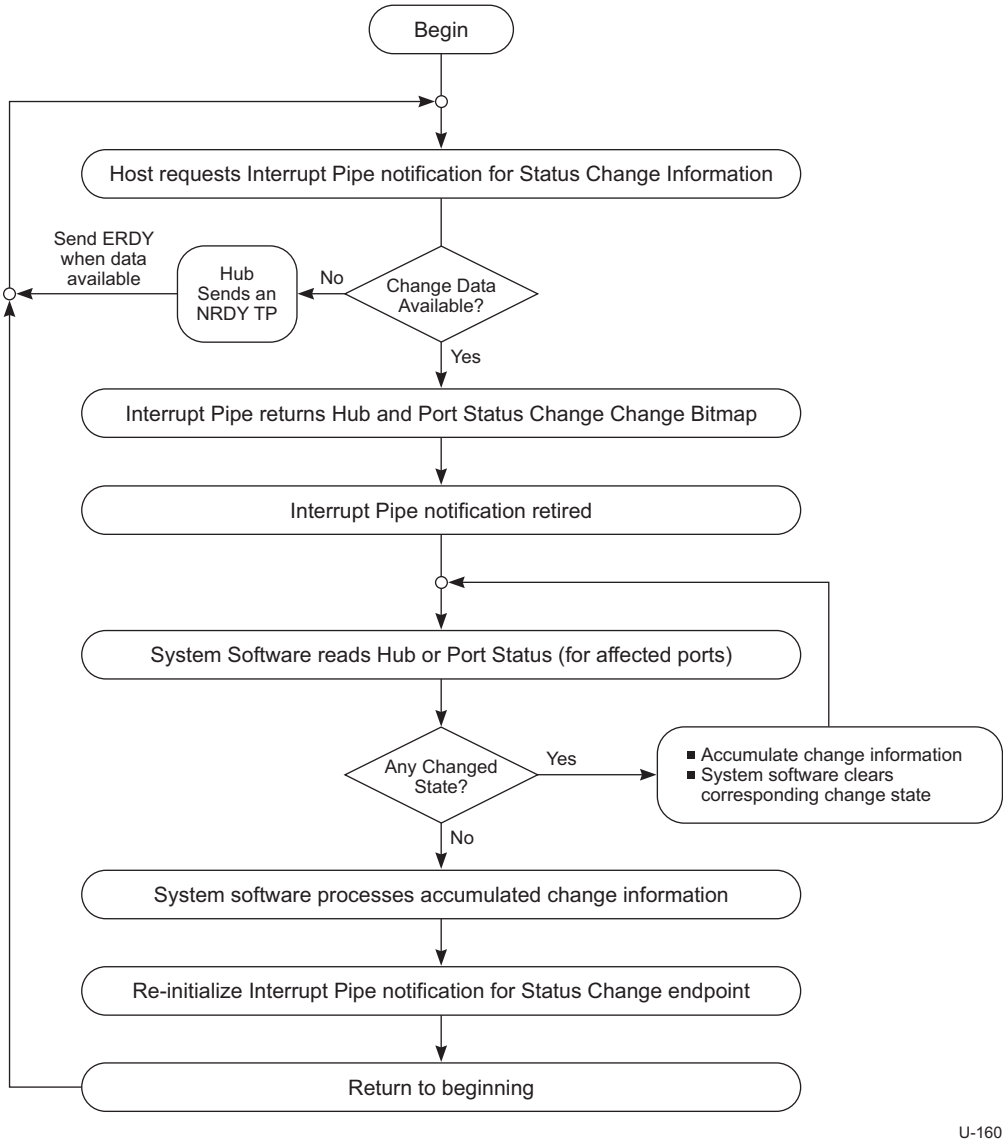


Figure 10-22. Port Status Handling Method

10.11.4 Hub and Port Status Change Bitmap

The Hub and Port Status Change Bitmap, shown in Figure 10-23, indicates whether the hub or a port has experienced a status change. This bitmap also indicates which port(s) have had a change in status. The hub returns this value on the Status Change endpoint. Hubs report this value in byte-increments. For example, if a hub has six ports, it returns a byte quantity, and reports a zero in the invalid port number field locations. The USB system software is aware of the number of ports on a hub (this is reported in the hub descriptor) and decodes the Hub and Port Status Change Bitmap accordingly. The hub reports any changes in hub status in bit zero of the Hub and Port Status Change Bitmap.

The Hub and Port Status Change Bitmap size is two bytes. Hubs report only as many bits as there are ports on the hub. A USB 3.0 hub may have no more than nMaxHubPorts.

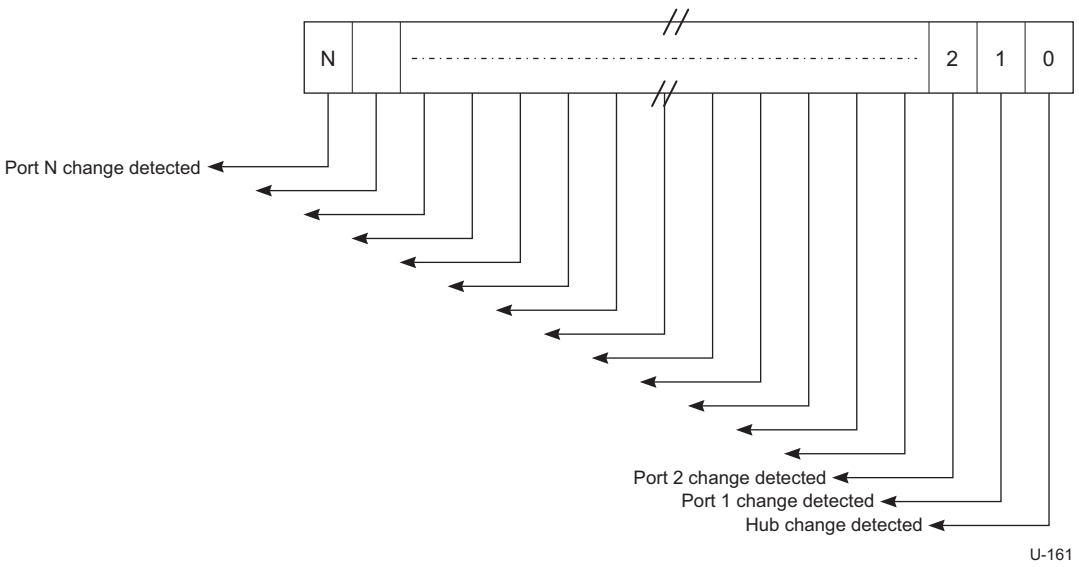


Figure 10-23. Hub and Port Status Change Bitmap

Any time any of the Status Changed bits are non-zero, an ERDY is returned (if an NRDY was previously sent) notifying the host that the Hub and Port Status Change Bitmap has changed. Figure 10-24 shows an example creation mechanism for hub and port change bits.

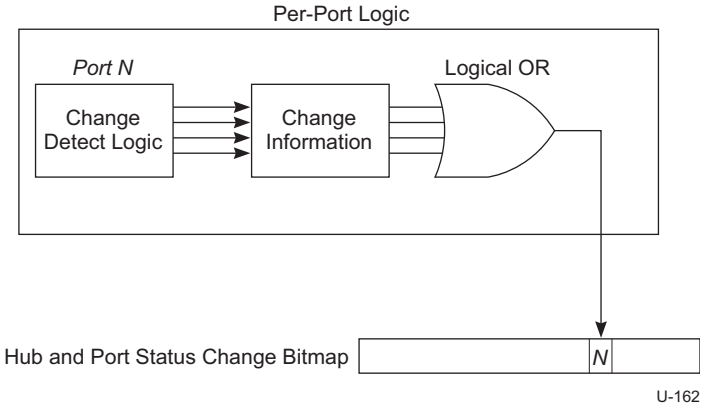


Figure 10-24. Example Hub and Port Change Bit Sampling

10.11.5 Over-current Reporting and Recovery

USB devices shall be designed to meet applicable safety standards. Usually, this will mean that a self-powered hub implements current limiting on its downstream facing ports. If an over-current condition occurs, it causes a status and state change in one or more ports. This change is reported to the USB system software so that it can take corrective action.

A hub may be designed to report over-current as either a port or a hub event. The hub descriptor field *wHubCharacteristics* is used to indicate the reporting capabilities of a particular hub (refer to Section 10.13.2.1). The over-current status bit in the hub or port status field indicates the state of the over-current detection when the status is returned. The over-current status change bit in the Hub or Port Change field indicates if the over-current status has changed.

When a hub experiences an over-current condition, it shall place all affected ports in the DSPORT.Powered-off-reset state. If a hub has per-port power switching and per-port current limiting, an over-current condition on one port may still cause the power on another port to fall below specified minimums. In this case, the affected port is placed in the DSPORT.Powered-off-reset state and C_PORT_OVER_CURRENT is set for the port, but PORT_OVER_CURRENT is not set. If the hub has over-current detection on a hub basis, then an over-current condition on the hub will cause all ports to enter any DSPORT.Powered-off-reset state. However, in this case, neither C_PORT_OVER_CURRENT nor PORT_OVER_CURRENT is set for the affected ports.

Host recovery actions for an over-current event should include the following:

1. Host gets change notification from hub with over-current event.
2. Host extracts appropriate hub or port change information (depending on the information in the change bitmap).
3. Host waits for over-current status bit to be cleared to 0.

4. Host cycles power to on for all of the necessary ports (e.g., issues a SetPortFeature(PORT_POWER) request for each port).
5. Host re-enumerates all affected ports.

10.11.6 Enumeration Handling

The hub device class commands are used to manipulate its downstream facing port state. When a device is attached, the device attach event is detected by the hub and reported on the Status Change endpoint. The host will accept the status change report and may request a SetPortFeature(PORT_RESET) on the port. The GetPortStatus request invoked by the host will return a PORT_CONNECTION indication along with the PORT_SPEED field set to zero if the downstream facing port has a SuperSpeed device connected.

When the device is detached from the port, the port reports the status change through the Status Change endpoint. Then the process is ready to be repeated on the next device attach detect.

10.12 Hub Configuration

Hubs are configured through the standard USB device configuration commands. A hub that is not configured behaves like any other device that is not configured with respect to power requirements and addressing. A hub is required to power its downstream ports based on several factors, including whether the hub supports power switching and charging applications. Refer to Section 10.3.1.1 for details on when a hub is required to provide power to downstream ports. Configuring a hub enables the Status Change endpoint. Part of the configuration process is setting the hub depth which is used to compute an index (refer to Section 10.14.2.8) into the Route String (refer to Section 8.9). The hub depth is used to derive the offset into the Route String (in a TP or DP) that the hub shall use to route packets received on its upstream port. The USB system software may then issue commands to the hub to switch port power on and off at appropriate times.

The USB system software examines hub descriptor information to determine the hub's characteristics. By examining the hub's characteristics, the USB system software ensures that illegal power topologies are not allowed by not powering on the hub's ports if doing so would violate the USB power topology. The device status and configuration information can be used to determine whether the hub can be used in a given topology. Table 10-3 summarizes the information and how it can be used to determine the current power requirements of the hub.

Table 10-3. Hub Power Operating Mode Summary

Configuration Descriptor		Hub Device Status (Self Power)	Explanation
MaxPower	bmAttributes (Self Powered)		
0	0	N/A	N/A This is an illegal combination.
0	1	0	N/A A device which is only self-powered, but does not have local power, cannot connect to the bus and communicate.

Configuration Descriptor		Hub Device Status (Self Power)	Explanation
MaxPower	bmAttributes (Self Powered)		
0	1	1	Self-powered only hub and local power supply is good. Hub status also indicates local power good. Hub functionality is valid anywhere depth restriction is not violated.
>0	0	N/A	Bus-powered only hub. Downstream facing ports may not be powered unless allowed in current topology. Hub device status reporting self-powered is meaningless if *bmAttributes.self-powered* is zero.
>0	1	0	This hub is capable of both self- and bus-powered operating modes. It is currently only available as a bus-powered hub.
> 0	1	1	This hub draws power from both the bus and its local power supply. It is currently available as a self-powered hub.

A self-powered hub has a local power supply, but may optionally draw one unit load from its upstream connection. This allows the interface to function when local power is not available (refer to Section 11.4.1.1). When local power is removed (either a hub-wide over-current condition or local supply is off), a hub of this type remains in the Configured state but transitions all ports (whether removable or non-removable) to the Powered-off state. While local power is off, all port status and change information read as zero and all SetPortFeature() requests are ignored (request is treated as a no-operation). The hub will use the Status Change endpoint to notify the USB system software of the hub event (refer to Section 10.11.4 for details on hub status).

The *MaxPower* field in the configuration descriptor is used to report to the system the maximum power the hub will draw from VBUS when the configuration is selected. The external devices attaching to the hub will report their individual power requirements.

A compound device may power both the hub electronics and the permanently attached devices from VBUS. The entire load may be reported in the hubs' configuration descriptor with the permanently attached devices each reporting self-powered, with zero *MaxPower* in their respective configuration descriptors.

A bus powered hub shall be able to supply any power not used by the hub electronics or permanently attached devices for the selected configuration to the exposed downstream ports. The hub shall be able to provide the power with any split across the exposed downstream ports (i.e., if the hub can provide 600 mA to two exposed downstream ports, it must be able to provide 450 mA to one and 150 mA to the other, 300 mA to each, etc.).

Note: Software shall ensure that at least 150 mA is available for each exposed downstream port on a bus powered hub.

10.13 Descriptors

Hub descriptors are derived from the general USB device framework. Hub descriptors describe a hub device and the ports on that hub. The host accesses hub descriptors through the hub's default control pipe.

The USB specification (refer to Chapter 8) defines the following descriptors:

- Device Level Descriptors
- Configuration
- Interface
- Endpoint
- String (optional)

The hub class defines additional descriptors (refer to Section 10.13.2). In addition, vendor-specific descriptors are allowed in the USB device framework. Hubs support standard USB device commands as defined in Chapter 8.

A hub is the only device that is allowed to function at high-speed and SuperSpeed at the same time. This specification only defines the descriptors a hub shall report when it is operating in SuperSpeed mode.

Note that a SuperSpeed hub shall always support the Get Descriptor (BOS) (refer to Section 9.6.2) in both SuperSpeed and non-SuperSpeed modes.

10.13.1 Standard Descriptors for Hub Class

The hub class pre-defines certain fields in standard USB descriptors. Other fields are either implementation-dependent or not applicable to this class.

A hub has a device descriptor with a bDeviceProtocol field set to 3 and an interface descriptor with a bInterfaceProtocol field set to 0.

Hub Descriptors in SuperSpeed Mode

Device Descriptor (SuperSpeed information)

bLength	18
bDescriptorType	1
bcdUSB	300H
bDeviceClass	HUB_CLASSCODE (9)
bDeviceSubClass	0
bDeviceProtocol	3
bMaxPacketSize0	9
bNumConfigurations	1

BOS Descriptor

bLength	5
bDescriptorType	BOS Descriptor type
wTotalLength	42
bNumDeviceCaps	3

USB 2.0 Extension

bLength	7
bDescriptorType	DEVICE CAPABILITY Descriptor type
bDevCapabilityType	2
bmAttributes	2

SuperSpeed USB Device Capability

bLength	10
bDescriptorType	DEVICE CAPABILITY Descriptor type
bDevCapabilityType	3
bmAttributes	Implementation-dependent
wSpeedsSupported	14
bFunctionalitySupport	1
bU1DevExitLat	Implementation-dependent
wU2DevExitLat	Implementation-dependent

ContainerID

bLength	20
bDescriptorType	DEVICE CAPABILITY Descriptor type
bDevCapabilityType	4
bReserved	0
ContainerID	Implementation-dependent

Configuration Descriptor (SuperSpeed information)

bLength	9
bDescriptorType	2
wTotalLength	31
bNumInterfaces	1
bConfigurationValue	X
iConfiguration	Y
bmAttributes	Z
bMaxPower	The maximum amount of bus power the hub will consume in this configuration

Interface Descriptor

bLength	9
bDescriptorType	4
bInterfaceNumber	0
bAlternateSetting	0
bNumEndpoints	1
bInterfaceClass	HUB_CLASSCODE (9)
bInterfaceSubClass	0
bInterfaceProtocol	0
iInterface	I

Endpoint Descriptor (for Status Change Endpoint)

bLength	7
bDescriptorType	5
bEndpointAddress	Implementation-dependent; Bit 7: Direction = In(1)
bmAttributes	Transfer Type = Interrupt (19)
wMaxPacketSize	2
bInterval	8 (maximum allowable interval)

Endpoint Companion Descriptor (for Status Change Endpoint)

bLength	6
bDescriptorType	48
bMaxBurst	0
bmAttributes	0
wBytesPerInterval	2

10.13.2 Class-specific Descriptors

10.13.2.1 Hub Descriptor

Table 10-4 outlines the various fields contained in the hub descriptor.

Table 10-4. SuperSpeed Hub Descriptor

Offset	Field	Size	Description
0	*bDescLength*	1	Number of bytes in this descriptor, including this byte. (12 bytes)
1	*bDescriptorType*	1	Descriptor Type, value: 2AH for SuperSpeed hub descriptor
2	*bNbrPorts*	1	Number of downstream facing ports that this hub supports. The maximum number of ports of ports a hub can support is 15.
3	*wHubCharacteristics*	2	D1...D0: Logical Power Switching Mode 00: Ganged power switching (all ports' power at once) 01: Individual port power switching 1X: Reserved D2: Identifies a Compound Device 0: Hub is not part of a compound device. 1: Hub is part of a compound device. D4...D3: Over-current Protection Mode 00: Global Over-current Protection. The hub reports over-current as a summation of all ports' current draw, without a breakdown of individual port over-current status. 01: Individual Port Over-current Protection. The hub reports over-current on a per-port basis. Each port has an over-current status. 1X: No Over-current Protection. This option is allowed only for bus-powered hubs that do not implement over-current protection. D15...D5: Reserved
5	*bPwrOn2PwrGood*	1	Time (in 2-ms intervals) from the time the power-on sequence begins on a port until power is good on that port. The USB system software uses this value to determine how long to wait before accessing a powered-on port. This value is set to zero if power-switching is not supported by the hub.
6	*bHubContrCurrent*	1	Maximum current requirements of the Hub Controller electronics when the hub is operating on both USB 2.0 and SuperSpeed expressed in units of aCurrentUnit (i.e., 50 = 50* aCurrentUnit mA). Note that the encoding of this field is different if the encoding used is the USB 2.0 specification for USB 2.0 hubs. A USB 3.0 hub shall report the current requirements when it is only operating on USB 2.0 (not SuperSpeed) in the USB 2.0 hub descriptor.

Offset	Field	Size	Description
7	*bHubHdrDecLat*	1	Hub Packet Header Decode Latency. Worst case latency for hubs whose upstream link is in U0 to decode the header of a downstream flowing TP or DP packet and initiate a transition to U0 on the relevant downstream port. The time is measured from receipt of the last symbol of the header packet by the upstream port until the hubs starts LFPS on the intended downstream port. This field is used to calculate the total path exit latency through a hub. The following are permissible values: Value — Meaning 00H — Much less than 0.1 µs. 01H — 0.1 µs 02H — 0.2 µs 03H — 0.3 µs 04H — 0.4 µs 05H — 0.5 µs 06H — 0.6 µs 07H — 0.7 µs 08H — 0.8 µs 09H — 0.9 µs 0AH — 1.0 µs 0BH - FFH — Reserved
8	*wHubDelay*	2	This field defines the average delay in nanoseconds a hub introduces while forwarding packets in either direction. The time is measured from receipt of the last symbol of the packet by the receiving port until the transmitting port sends the first framing symbol of the packet, when both the receiving and transmitting links are in U0 and the following conditions are met: • No Link Commands or SKP ordered sets or other packets are in flight. • Remote Rx Header Buffer Credit Count of the transmitting port is not zero. • Tx Header Buffer of the transmitting port is empty. Note that the maximum value a hub is allowed to report in this field is tHubDelay.

Offset	Field	Size	Description
10	*DeviceRemovable*	2	Indicates if a port has a removable device attached. This field is reported on byte-granularity. Within a byte, if no port exists for a given location, the bit field representing the port characteristics shall be 0. Bit value definition: 0B - Device is removable. 1B - Device is non-removable. This is a bitmap corresponding to the individual ports on the hub: Bit 0: Reserved for future use Bit 1: Port 1 Bit 2: Port 2 .... Bit *n*: Port *n* (implementation-dependent, up to a maximum of 15 ports)

10.14 Requests

10.14.1 Standard Requests

Hubs have tighter constraints on request processing timing than specified in Section 9.2.6 for standard devices because they are crucial to the "time to availability" of all devices attached to the USB. The worst case request timing requirements are listed below (they apply to both Standard and Hub Class requests):

- Completion time for requests with no data stage: 50 ms
- Completion times for standard requests with data stage(s):
  - Time from setup packet to first data stage: 50 ms
  - Time between each subsequent data stage: 50 ms
  - Time between last data stage and status stage: 50 ms

Because hubs play such a crucial role in bus enumeration, it is recommended that hubs average response times be less than 5 ms for all requests.

Table 10-5 outlines the various standard device requests.

Table 10-5. Hub Responses to Standard Device Requests

bRequest	Hub Response
CLEAR_FEATURE	Standard
GET_CONFIGURATION	Standard
GET_DESCRIPTOR	Standard
GET_INTERFACE	Undefined. Hubs are allowed to support only one interface.
GET_STATUS	Standard
SET_ADDRESS	Standard
SET_CONFIGURATION	Standard
SET_DESCRIPTOR	Optional
SET_FEATURE	Standard
SET_INTERFACE	Undefined. Hubs are allowed to support only one interface.
SET_ISOCH_DELAY	Standard
SET_SEL	Standard
SYNCH_FRAME	Undefined. Hubs are not allowed to have isochronous endpoints.

A hub is required to accept all "Standard" requests without error. A hub shall not respond with a request error to a well-formed SET_ISOC_DELAY request. A hub is not required to retain or process the delay value. Optional requests that are not implemented shall return a STALL in the Data stage or Status stage of the request.

10.14.2 Class-specific Requests

The hub class defines requests to which hubs respond, as outlined in Table 10-6. Table 10-7 defines the hub class request codes. All requests in the table below except SetHubDescriptor() are mandatory.

Table 10-6. Hub Class Requests

Request	bmRequestType	bRequest	wValue	wIndex	wLength	Data
ClearHubFeature	00100000B	CLEAR_FEATURE	Feature Selector	Zero	Zero	None
ClearPortFeature	00100011B	CLEAR_FEATURE	Feature Selector	Port	Zero	None
GetHubDescriptor	10100000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
GetHubStatus	10100000B	GET_STATUS	Zero	Zero	Four	Hub Status and Change Status
GetPortStatus	10100011B	GET_STATUS	Zero	Port	Four	Port Status and Change Status
GetPortErrorCount	10100011B	GET_PORT_ERR_ COUNT	Zero	Port	Two	Number of Link Errors on this port
SetHubDescriptor	00100000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
SetHubFeature	00100000B	SET_FEATURE	Feature Selector	Zero	Zero	None
SetHubDepth	00100000B	SET_HUB_DEPTH	Hub Depth	Zero	Zero	None
SetPortFeature	00100011B	SET_FEATURE	Feature Selector	Selector, Timeout, Port	Zero	None

Table 10-7. Hub Class Request Codes

bRequest	Value
GET_STATUS	0
CLEAR_FEATURE	1
RESERVED (used in previous specifications for GET_STATE)	2
SET_FEATURE	3
RESERVED	*4-5*
GET_DESCRIPTOR	6
SET_DESCRIPTOR	7
RESERVED (used in USB 2.0 specification)	8-11
SET_HUB_DEPTH	12
GET_PORT_ERR_COUNT	13

Table 10-8 gives the valid feature selectors for the hub class. Refer to Section 10.14.2.4 and Section 10.14.2.6 for a description of the features.

Table 10-8. Hub Class Feature Selectors

Feature Selector	Recipient	Value
C_HUB_LOCAL_POWER	Hub	0
C_HUB_OVER_CURRENT	Hub	1
PORT_CONNECTION	Port	0
PORT_OVER_CURRENT	Port	3
PORT_RESET	Port	4
PORT_LINK_STATE	Port	5
PORT_POWER	Port	8
C_PORT_CONNECTION	Port	16
C_PORT_OVER_CURRENT	Port	19
C_PORT_RESET	Port	20
RESERVED (used in USB 2.0 specification)	Port	21
PORT_U1_TIMEOUT	Port	23
PORT_U2_TIMEOUT	Port	24
C_PORT_LINK_STATE	Port	25
C_PORT_CONFIG_ERROR	Port	26
PORT_REMOTE_WAKE_MASK	Port	27
BH_PORT_RESET	Port	28
C_BH_PORT_RESET	Port	29
FORCE_LINKPM_ACCEPT	Port	30

10.14.2.1 Clear Hub Feature

This request resets a value reported in the hub status.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B	CLEAR_FEATURE	Feature Selector	Zero	Zero	None

Clearing a feature disables that feature; refer to Table 10-8 for the feature selector definitions that apply to the hub as a recipient. If the feature selector is associated with a status change, clearing that status change acknowledges the change. This request format is used to clear either the C_HUB_LOCAL_POWER or C_HUB_OVER_CURRENT features.

It is a Request Error if *wValue* is not a feature selector listed in Table 10-8 or if *wIndex* or *wLength* are not as specified above.

If the hub is not configured, the hub's response to this request is undefined.

10.14.2.2 Clear Port Feature

This request resets a value reported in the port status.

bmRequestType	bRequest	wValue	wIndex		wLength	Data
00100011B	CLEAR_FEATURE	Feature Selector	Selector	Port	Zero	None

The port number shall be a valid port number for that hub, greater than zero. The port field is located in bits 7..0 of the *wIndex* field.

Clearing a feature disables that feature or starts a process associated with the feature; refer to Table 10-8 for the feature selector definitions. If the feature selector is associated with a status change, clearing that status change acknowledges the change. This request format is used to clear the following features:

- PORT_POWER
- C_PORT_CONNECTION
- C_PORT_RESET
- C_PORT_OVER_CURRENT
- C_PORT_LINK_STATE
- C_PORT_CONFIG_ERROR
- C_BH_PORT_RESET

Clearing the PORT_POWER feature causes the port to be placed in the DSPORT.Powered-off-reset state and may, subject to the constraints due to the hub's method of power switching, result in power being removed from the port. When in the DSPORT.Powered-off or the DSPORT.Powered-off-detect or the DSPORT.Powered-off-reset state, the only requests that are valid when this port is the recipient are Get Port Status (refer to Section 10.14.2.6) and Set Port Feature (PORT_POWER) (refer to Section 10.14.2.10).

Clearing the FORCE_LINKPM_ACCEPT feature causes the port to de-assert the Force_LinkPM_Accept bit in Set Link Function LMPs. If the Force_LinkPM_Accept bit is not asserted on the port, the hub shall treat this request as a functional no-operation.

It is a Request Error if *wValue* is not a feature selector listed in Table 10-8, if *wIndex* specifies a port that does not exist, or if *wLength* is not as specified above. It is not an error for this request to try to clear a feature that is already cleared (the hub shall treat this as a functional no-operation).

If the hub is not configured, the hub's response to this request is undefined.

10.14.2.3 Get Hub Descriptor

This request returns the hub descriptor.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero	Descriptor Length	Descriptor

The GetDescriptor() request for the hub class descriptor follows the same usage model as that of the standard GetDescriptor() request (refer to Chapter 9). The standard hub descriptor is denoted by using the value *bDescriptorType* defined in Section 10.13.2.1. All hubs are required to implement one hub descriptor, with descriptor index zero.

If *wLength* is larger than the actual length of the descriptor, then only the actual length is returned. If *wLength* is less than the actual length of the descriptor, then only the first *wLength* bytes of the descriptor are returned; this is not considered an error even if *wLength* is zero.

It is a Request Error if *wValue* or *wIndex* are other than as specified above.

If the hub is not configured, the hub's response to this request is undefined.

10.14.2.4 Get Hub Status

This request returns the current hub status and the states that have changed since the previous acknowledgment.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100000B	GET_STATUS	Zero	Zero	Four	Hub Status and Change Status

The first word of data contains the wHubStatus field (refer to Table 10-9). The second word of data contains the wHubChange field (refer to Table 10-10).

It is a Request Error if *wValue*, *wIndex*, or *wLength* are other than as specified above.

If the hub is not configured, the hub's response to this request is undefined.

Table 10-9. Hub Status Field, *wHubStatus*

Bit	Description
0	Local Power Source: This is the source of the local power supply. This field indicates whether hub power (for other than the SIE) is being provided by an external source or from the USB. This field allows the USB system software to determine the amount of power available from a hub to downstream devices. 0 = Local power supply good 1 = Local power supply lost (inactive)
1	Over-current: If the hub supports over-current reporting on a hub basis, this field indicates that the sum of all the ports' current has exceeded the specified maximum and all ports have been placed in the Powered-off-reset state. If the hub reports over-current on a per-port basis or has no over-current detection capabilities, this field is always zero. The hub shall only report over-current if it is physically unable to meet the sum of all ports' current draws. For more details on over-current protection, refer to the USB 2.0 Specification, Section 7.2.1.2.1. 0 = No over-current condition currently exists. 1 = A hub over-current condition exists.
2-15	Reserved

There are no defined feature selector values for these status bits and they can neither be set nor cleared by the USB system software.

Table 10-10. Hub Change Field, *wHubChange*

Bit	Description
0	Local Power Status Change (C_HUB_LOCAL_POWER): This field indicates that a change has occurred in the hub's Local Power Source field in *wHubStatus*. This field is initialized to zero when the hub receives a bus reset. 0 = No change has occurred to Local Power Status. 1 = Local Power Status has changed.
1	Over-Current Change (C_HUB_OVER_CURRENT): This field indicates if a change has occurred in the Over-Current field in *wHubStatus*. This field is initialized to zero when the hub receives a bus reset. 0 = No change has occurred to the Over-Current Status. 1 = Over-Current Status has changed.
2-15	Reserved

Hubs may allow setting of these change bits with SetHubFeature() requests in order to support diagnostics. If the hub does not support setting of these bits, it shall either treat the SetHubFeature() request as a Request Error or as a functional no-operation. When set, these bits may be cleared by a ClearHubFeature() request. A request to set a feature that is already set or to clear a feature that is already clear has no effect and the hub shall treat this as a functional no-operation.

10.14.2.5 Get Port Error Count

This request returns the number of link errors detected by the hub on the port indicated by wIndex. This value is reset to zero whenever the device goes through a Reset (refer to Section 7.3) or at power up.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100011B	GET_PORT_ERR_COUNT	Zero	Port	Two	Number of Link Errors

The port number shall be a valid port number for that hub, greater than zero.

It is a Request Error if *wValue* or *wLength* are other than as specified above or if *wIndex* specifies a port that does not exist.

If the hub is not configured, the behavior of the hub in response to this request is undefined.

10.14.2.6 Get Port Status

This request returns the current port status and the current value of the port status change bits.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100011B	GET_STATUS	Zero	Port	Four	Port Status and Change Status

The port number shall be a valid port number for that hub, greater than zero.

The first word of data contains the wPortStatus field (refer to Table 10-11). The second word of data contains the wPortChange field (refer to Table 10-12).

The bit locations in the wPortStatus and wPortChange fields correspond in a one-to-one fashion where applicable.

It is a Request Error if *wValue* or *wLength* are other than as specified above or if *wIndex* specifies a port that does not exist.

If the hub is not configured, the behavior of the hub in response to this request is undefined.

10.14.2.6.1 Port Status Bits

Table 10-11. Port Status Field, *wPortStatus*

Bit	Description
0	Current Connect Status (PORT_CONNECTION): This field reflects whether or not a device is currently connected to this port. **Value** **Meaning** 0 No device is present 1 A device is present on this port
1	Port Enabled/Disabled: This field indicates whether the port is enabled. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the USB system software. **Value** **Meaning** 0 Port is disabled 1 Port is enabled
2	Reserved
3	Over-current (PORT_OVER_CURRENT): If the hub reports over-current conditions on a per-port basis, this field indicates that the current drain on the port exceeds the specified maximum. **Value** **Meaning** 0 No over-current condition exists on this port 1 An over-current condition exists on this port
4	Reset (PORT_RESET): This field is set when the host wishes to reset the attached device. It remains set until the reset signaling is turned off by the hub. **Value** **Meaning** 0 Reset signaling not asserted 1 Reset signaling asserted
5-8	Port Link State (PORT_LINK_STATE): This field reflects the current state of the link attached to this port. A new state is not reflected until the link state transition to that state is complete. **Value** **Meaning** 0x00 Link is in the U0 State 0x01 Link is in the U1 State 0x02 Link is in the U2 State 0x03 Link is in the U3 State 0x04 Link is in the SS.Disabled State 0x05 Link is in the Rx.Detect State 0x06 Link is in the SS.Inactive State 0x07 Link is in the Polling State 0x08 Link is in the Recovery State 0x09 Link is in the Hot Reset State 0xA Link is in the Compliance Mode State 0xB Link is in the Loopback State 0xC-0xF Reserved

Bit	Description
9	Port Power (PORT_POWER): This field reflects a port's logical, power control state. Because hubs can implement different methods of port power switching, this field may or may not represent whether power is applied to the port. The device descriptor reports the type of power switching implemented by the hub. **Value** **Meaning** 0 This port is in the Powered-off state 1 This port is not in the Powered-off state
10-12	Negotiated speed of the SuperSpeed Device Attached to this port (PORT_SPEED): This field is valid only if a device is attached. **Value** **Meaning** 0 5 Gbps 1 Reserved 2 Reserved 3 Reserved 4 Reserved 5 Reserved 6 Reserved 7 Reserved
13-15	Reserved

PORT_CONNECTION

This bit is set to one when the port in the DSPORT.Enabled state. In DSPORT.Resetting or DSPORT.Error state it maintains the value from prior state.

SetPortFeature(PORT_CONNECTION) and ClearPortFeature(PORT_CONNECTION) requests shall not be used by the USB system software and shall be treated as no-operation requests by hubs.

PORT_ENABLE

This bit is set to one when the downstream port is in the DSPORT.Enabled state and is set to zero otherwise.

Note that the USB 2.0 ClearPortFeature (PORT_ENABLE) request is not supported by SuperSpeed hubs and cannot be used by USB system software to disable a port.

PORT_OVER_CURRENT

This bit is set to one while an over-current condition exists on the port and set to zero otherwise.

If the voltage on this port is affected by an over-current condition on another port, this bit is set to one and remains set to one until the over-current condition on the affecting port is removed. When the over-current condition on the affecting port is removed, this bit is set to zero.

Over-current protection is required on self-powered hubs (it is optional on bus-powered hubs) as outlined in Section 10.10.

The SetPortFeature(PORT_OVER_CURRENT) and ClearPortFeature(PORT_OVER_CURRENT) requests shall not be used by the USB system software and may be treated as no-operation requests by hubs.

PORT_RESET

This bit is set to one while the port is in the DSPORT.Resetting state. This bit is set to zero in all other downstream port states.

A SetPortFeature(PORT_RESET or BH_PORT_RESET) request will initiate the DSPORT.Resetting state if the conditions in Section 10.3.1.6 are met.

The ClearPortFeature(PORT_RESET) request shall not be used by the USB system software and may be treated as a no-operation request by hubs.

PORT_LINK_STATE

This field reflects the current state of the link.

The SetPortFeature(PORT_LINK_STATE) request may be issued by the USB system software at any time but will have an effect only as specified in Section 10.14.2.10.

The ClearPortFeature(PORT_LINK_STATE) requests shall not be used by the USB System software and may be treated as no-operation requests by hubs.

PORT_POWER

This bit reflects the current logical power state of a port. This bit is implemented on all ports whether or not actual port power switching devices are present.

While this bit is zero, the port is in the DSPORT.Powered-off state, the DSPORT.Powered-off-detect state, or the DSPORT.Powered-off-reset state. Similarly, anything that causes this port to go to any of these three states will cause this bit to be set to zero.

A SetPortFeature(PORT_POWER) will set this bit to one unless both C_HUB_LOCAL_POWER and Local Power Status (in *wHubStatus*) are set to one in which case the request is treated as a functional no-operation.

PORT_SPEED

This value in this field is only valid when the PORT_ENABLE bit is set to one. A value of zero in this field indicates that the SuperSpeed device attached to this port is operating at 5 Gbps. All other values in this field are reserved.

This field can only be read by USB system software.

10.14.2.6.2 Port Status Change Bits

Port status change bits are used to indicate changes in port status bits that are not the direct result of requests. Port status change bits can be cleared with a ClearPortFeature() request or by a hub reset. Hubs may allow setting of the status change bits with a SetPortFeature() request for diagnostic purposes. If a hub does not support setting of the status change bits, it may either treat the request as a Request Error or as a functional no-operation. Table 10-12 describes the various bits in the *wPortChange* field.

Table 10-12. Port Change Field, *wPortChange*

Bit	Description
0	Connect Status Change (C_PORT_CONNECTION): Indicates a change has occurred in the port's Current Connect Status. The hub device sets this field as described in Section 10.3.1. **Value** **Meaning** 0 No change has occurred to Current Connect status 1 Current Connect status has changed
1-2	Reserved
3	Over-Current Indicator Change (C_PORT_OVER_CURRENT): This field applies only to hubs that report over-current conditions on a per-port basis (as reported in the hub descriptor). **Value** **Meaning** 0 No change has occurred to Over-Current Indicator 1 Over-Current Indicator has changed If the hub does not report over-current on a per-port basis, then this field is always zero.
4	Reset Change (C_PORT_RESET): This field is set when reset processing for any type of reset on this port is complete. **Value** **Meaning** 0 No change 1 Reset complete
5	BH Reset Change (C_BH_PORT_RESET): This field is set when a warm reset processing on this port is complete **Value** **Meaning** 0 No change 1 Reset complete
6	Port Link State Change (C_PORT_LINK_STATE): This field is set when the port link status has changed as described below. **Value** **Meaning** 0 No change 1 Link Status has changed
7	Port Config Error (C_PORT_CONFIG_ERROR): This field is set when the port fails to configure its link partner. **Value** **Meaning** 0 Port Link Configuration was successful 1 Port Link Configuration was unsuccessful
8-15	Reserved

C_PORT_CONNECTION

This bit is set to one when the PORT_CONNECTION bit changes.

This bit shall be set to zero by a ClearPortFeature(C_PORT_CONNECTION) request or while logical port power is off.

C_PORT_OVER_CURRENT

This bit is set to one when the PORT_OVER_CURRENT bit changes from zero to one or from one to zero. This bit is also set if the port is placed in the DSPORT.Powered-off-reset state due to an over-current condition on another port.

This bit shall be set to zero by a ClearPortFeature(C_PORT_OVER_CURRENT) request.

C_PORT_RESET

This bit is set to one when the port transitions from the DSPORT.Resetting state to the DSPORT.Enabled state for any type of reset.

This bit shall be set to zero by a ClearPortFeature(C_PORT_RESET) request, or while logical port power is off.

C_PORT_BH_RESET

This bit is set to one when the port transitions from the DSPORT.Resetting state to the DSPORT.Enabled state for a Warm Reset only.

This bit shall be cleared by a ClearPortFeature(C_PORT_BH_RESET) request, or while logical port power is off.

C_PORT_LINK_STATE

This bit is set to one when the port's link completes a transition from the U3 state to the U0 state as a result of a SetPortFeature(Port_Link_State) request or completes a transition to Loopback state or to Compliance or from any of the U-states to SS.Inactive (with Rx terminations present). This bit is not set to one due to transitions from U3 to U0 as a result of remote wakeup signaling received on a downstream facing port.

This bit will be cleared by a ClearPortFeature(C_PORT_LINK_STATE) request, or while logical port power is off.

C_PORT_CONFIG_ERROR

This bit is set to one if the link connected to the port could not be successfully configured, e.g., if two downstream only capable ports are connected to each other or if the link configuration could not be completed. In addition, the port shall transition to the DSPORT.Error state when this occurs.

This bit will be cleared by a ClearPortFeature(C_PORT_CONFIG_ERROR) request, or while logical port power is off.

10.14.2.7 Set Hub Descriptor

This request overwrites the hub descriptor.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero	Descriptor Length	Descriptor

The SetDescriptor request for the hub class descriptor follows the same usage model as that of the standard SetDescriptor request (refer to the framework chapter). The standard hub descriptor is denoted by using the value *bDescriptorType* defined in Section 10.13.2.1. All hubs are required to implement one hub descriptor with descriptor index zero.

This request is optional. This request writes data to a class-specific descriptor. The host provides the data that is to be transferred to the hub during the data transfer phase of the control transaction. This request writes the entire hub descriptor at once.

Hubs shall buffer all the bytes received from this request to ensure that the entire descriptor has been successfully transmitted from the host. Upon successful completion of the bus transfer, the hub updates the contents of the specified descriptor.

It is a Request Error if *wIndex* is not zero or if *wLength* does not match the amount of data sent by the host. Hubs that do not support this request respond with a STALL during the Data stage of the request.

If the hub is not configured, the hub's response to this request is undefined.

10.14.2.8 Set Hub Feature

This request sets a value reported in the hub status.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B	SET_FEATURE	Feature Selector	Zero	Zero	None

Setting a feature enables that feature; refer to Table 10-8 for the feature selector definitions that apply to the hub as recipient. Status changes may not be acknowledged using this request.

It is a Request Error if *wValue* is not a feature selector listed in Table 10-8 or if *wIndex* or *wLength* are not as specified above.

If the hub is not configured, the hub's response to this request is undefined.

10.14.2.9 Set Hub Depth

This request sets the value that the hub uses to determine the index into the Route String Index for the hub.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B	SET_HUB_DEPTH	Hub Depth	Zero	Zero	None

wValue has the value of the Hub Depth. The Hub Depth left shifted by two is the offset into the Route String that identifies the lsb of the Route String Port Field for the hub.

It is a Request Error if *wValue* is greater than 4 or if *wIndex* or *wLength* are not as specified above.

If the hub is not configured, the hub's response to this request is undefined.

10.14.2.10 Set Port Feature

This request sets a value reported in the port status.

bmRequestType	bRequest	wValue	wIndex		wLength	Data
00100011B	SET_ FEATURE	Feature Selector	Selector or Timeout Value or Remote Wake Mask	Port	Zero	None

The port number shall be a valid port number for that hub, greater than zero. The port number is in the least significant byte (bits 7..0) of the *wIndex* field. The most significant byte of *wIndex* is zero, except when the feature selector is PORT_U1_TIMEOUT or PORT_U2_TIMEOUT or PORT_LINK_STATE or PORT_REMOTE_WAKE_MASK.

Setting a feature enables that feature or starts a process associated with that feature; see Table 10-8 for the feature selector definitions that apply to a port as a recipient. Status change may not be acknowledged using this request. Features that can be set with this request are:

- PORT_RESET
- BH_PORT_RESET
- PORT_POWER
- PORT_U1_TIMEOUT
- PORT_U2_TIMEOUT
- PORT_LINK_STATE
- PORT_REMOTE_WAKE_MASK
- FORCE_LINKPM_ACCEPT

When the feature selector is PORT_U1_TIMEOUT, the most significant byte (bits 15..8) of the *wIndex* field specifies the Timeout value for the U1 inactivity timer. Refer to Section 10.4.2.1 for a detailed description of how the U1 inactivity timer value is used.

The following are permissible values:

Table 10-13. U1 Timeout Value Encoding

Value	Description
00H	Zero (Default)
01H	1 µs
02H	2 µs
03H	3 µs
…	…
7FH	127 µs
80H-FEH	Reserved
FFH	Infinite

When the feature selector is PORT_U2_TIMEOUT, the most significant byte (bits 15..8) of the *wIndex* field specifies the Timeout value for the U2 inactivity timer. The port's link shall send an LMP to its link partner with the specified timeout value after receiving a Set Port Feature request with the PORT_U2_TIMEOUT feature selector. Refer to Section 10.4.2.1 for a detailed description of how the U2 inactivity timer value is used.

The following are permissible values:

Table 10-14. U2 Timeout Value Encoding

Value	Description
00H	Zero (Default)
01H	256 µs
02H	512 µs
03H	768 µs
…	…
FEH	65.024 ms
FFH	Infinite

Note: It is the responsibility of software to properly set the U2 timeout for a downstream port that is connected to a hub. Inconsistent link states could result if the timeout is not set properly. It is recommended that software should set the upstream U2 timeout to at least twice the value of the U2 timeout of the downstream ports on the hub.

When the feature selector is PORT_LINK_STATE, the most significant byte (bits 15..8) of the *wIndex* field specifies the U state the host software wants to put the link connected to the port into. This request is only valid when the PORT_ENABLE bit is set and the PORT_LINK_STATE is not set to SS.Disabled, Rx.Detect or SS.Inactive except as noted below:

- If the value is 0, then the hub shall transition the link to U0 from any of the U states.
- If the value is 1, then host software wants to transition the link to the U1 State. The hub shall attempt to transition the link to U1 from U0. If the link is in any state other than U0 when a request is received with a value of 1, the behavior is undefined.
- If the value is 2, then the host software wants to transition the link to the U2 State. The hub shall attempt to transition the link to U2 from U0. If the link is in any state other than U0 when a request is received with a value of 2, the behavior is undefined.

- If the value is 3, then host software wants to selectively suspend the device connected to this port. The hub shall transition the link to U3 from any of the other U states using allowed link state transitions. If the port is not already in the U0 state, then it shall transition the port to the U0 state and then initiate the transition to U3. While this state is active, the hub does not propagate downstream-directed traffic to this port, but the hub will respond to resume signaling from the port.
- If the value is 4 (SS.Disabled), the hub shall transition the link to SS.Disabled. The request is valid at all times when the value is 4. The downstream port shall transition to the DSPORT.Disabled state after this request is received.
- If the value is 5 (Rx.Detect), the hub shall transition the link to Rx.Detect. This request is only valid when the downstream port is in the DSPORT.Disabled state. If the link is in any other state when a request is received with this value, the behavior is undefined. The downstream port shall transition to the DSPORT.Disconnected state after this request is received.
- The hub shall respond with a Request Error if it sees any other value in the upper byte of the *wIndex* field.

When the feature selector is PORT_REMOTE_WAKE_MASK, the most significant byte (bits 15..8) of the *wIndex* field specifies the conditions that would cause the hub to signal a remote wake event on its upstream port. The encoding for the port remote wake mask is given below:

Table 10-15. Downstream Port Remote Wake Mask Encoding

Bit	Description
0	Conn_RWEnable **Value** — **Meaning** 0 — The hub is disabled from signaling a remote wakeup due to a connect event on this port; connect events that occur during suspend must still be detected and reported after the resume process has completed (due to some other event) as a C_PORT_CONNECTION port status change. 1 — The hub is enabled to signal a remote wakeup due to a connect event on the port and if Function Remote Wake is also enabled.
1	Disconn_RWEnable **Value** — **Meaning** 0 — The hub is disabled from signaling a remote wakeup due to a disconnect event on this port; disconnect events that occur during suspend must still be detected and reported after the resume process has completed (due to some other event) as a C_PORT_CONNECTION port status change. 1 — The hub is enabled to signal a remote wakeup due to a disconnect event on the port and if Function Remote Wake is also enabled.

Bit	Description
2	OC_RWEnable **Value** **Meaning** 0 The hub is disabled from signaling a remote wakeup due to an over-current event on this port; over-current events that occur during suspend must still be detected and reported after the resume process has completed (due to some other event) as a C_PORT_OVER_CURRENT port status change. Note that a hub that does not support per-port over current detection/reporting will signal remote-wakeup for an over-current event unless all ports have OC-RWEnable set to 0. 1 The hub is enabled to signal a remote wakeup due to an over-current event on the port and if Function Remote Wake is also enabled.
3-7	These bits are reserved and must be set to zero.

Note that after power on or after the hub is reset, the remote wake mask is set to zero (i.e., the mask is enabled).

The hub shall meet the following requirements:

- If the port is in the Powered-off state, the hub shall treat a SetPortFeature(PORT_RESET) request as a functional no-operation.
- If the port is not in the Enabled state, the hub shall treat a SetPortFeature(PORT_LINK_STATE) U3 request as a functional no-operation.
- If the port is not in the Powered-off state, the hub shall treat a SetPortFeature(PORT_POWER) request as a functional no-operation.
- If the port is not in the Enabled state, the hub shall treat a SetPortFeature(FORCE_LINKPM_ACCEPT) request as a functional no-operation.

When the feature selector is BH_PORT_RESET, the hub shall initiate a warm reset (refer to Section 7.3) on the port that is identified by this command. The state of the port after this reset shall be the same as the state after a SetPortFeature(PORT_RESET). On completion of a BH_PORT_RESET, the hub shall set the C_BH_PORT_RESET field to one in the PortStatus for this port.

It is a Request Error if *wValue* is not a feature selector listed in Table 10-8, if *wIndex* specifies a port that does not exist, or if *wLength* is not as specified above.

If the hub is not configured, the hub's response to this request is undefined.

10.15 Host Root (Downstream) Ports

The root ports of a USB 3.0 host have similar functional requirements to the downstream ports of a USB 3.0 hub. This section summarizes which requirements also apply to the root port of a host and identifies any additional or different requirements.

A host root port shall follow the requirements for a downstream facing hub port in Section 10.2 except for Section 10.2.3.

A host root port shall follow the requirements for a downstream facing hub port in Section 10.3 with the following exceptions and additions:

- None of the transitions and/or transition conditions based on the state of the hub upstream port apply to a root port.
- A host shall have control mechanisms in the host interface that allow software to achieve equivalent behavior to hub downstream port behavior in response to SetPortFeature or ClearPortFeature requests documented in Section 10.3.
- A host shall implement port status bits consistent with the downstream port state descriptions in Section 10.3.
- A host is required to provide a mechanism to correlate each USB 2.0 port with any SuperSpeed port that shares the same physical connector. Note that this is similar to the requirement for USB 3.0 hubs in Section 10.3.3.

A host root port shall follow the requirements for a downstream facing hub port in Section 10.4 with the same general exceptions already noted in this section.

A host shall implement port status bits through the host interface that are equivalent to all port status bit definitions in this chapter.

A host shall have mechanisms to achieve equivalent control over its root ports as provided by the SetPortFeature, ClearPortFeature, and GetPortStatus requests documented in this chapter.

10.16 Peripheral Device Upstream Ports

The upstream port of a peripheral device has similar functional requirements to the upstream port of a USB 3.0 hub. This section summarizes which requirements also apply to the upstream port of a peripheral device and identifies any additional or different requirements.

10.16.1 Peripheral Device Upstream Ports

A peripheral device shall follow the requirements for an upstream facing hub port in Section 10.5 with the following exceptions and additions:

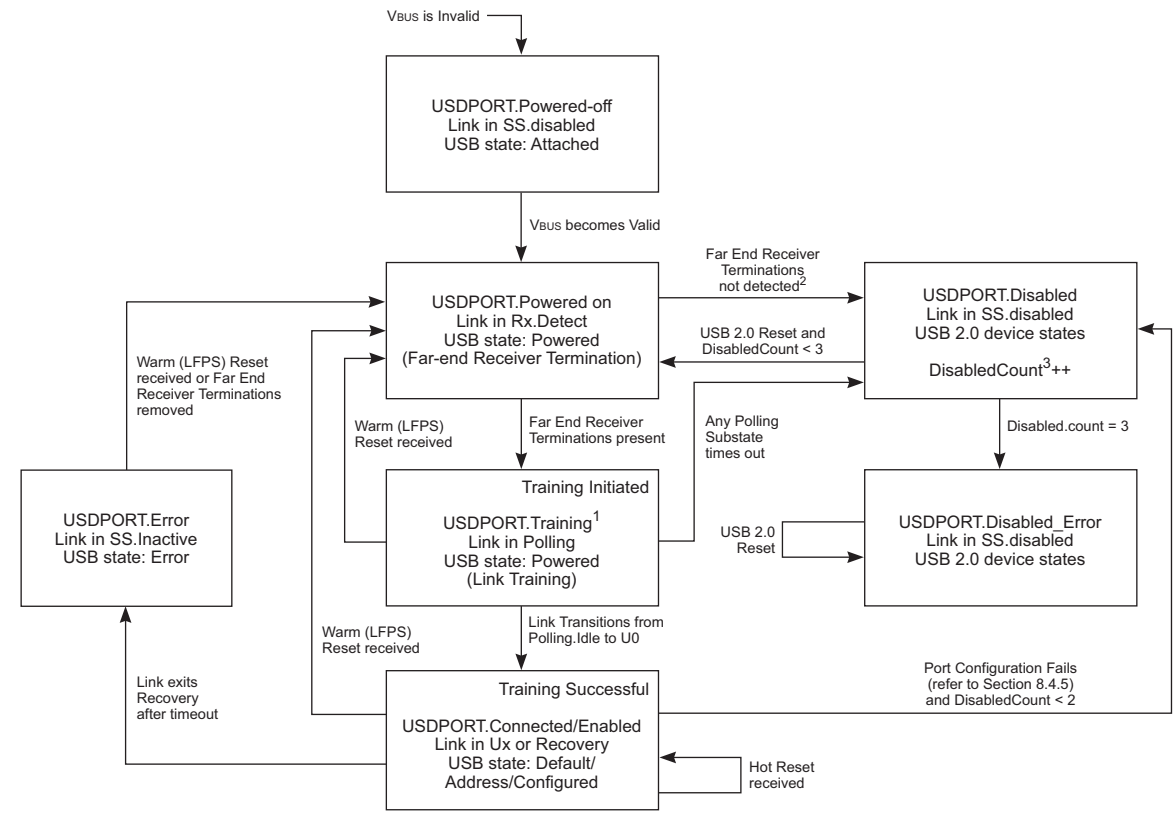
- A peripheral device shall not attempt to connect on the USB 2.0 interface when the port is in the USPORT.Connected state.
- A peripheral device shall not attempt to connect on the USB 2.0 interface unless the port has entered the USPORT.Powered-off state and V_{BUS} is still present as shown in Figure 10-11.
- If a device is connected on the USB 2.0 interface and it receives a USB 2.0 bus reset, the device shall enter the USPORT.Powered-On state within tCheckSuperSpeedOnReset time.
- After a USB 2.0 reset, if the SuperSpeed port enters the USPORT.Training state, the device shall disconnect on the USB 2.0 interface within tUSB2SwitchDisconnect time.

A device shall follow the requirements for an upstream facing hub port in Section 10.6 with the following exceptions and additions:

- None of the conditions related to downstream port apply.
- A peripheral device initiates transitions to U1 or U2 when otherwise allowed based on vendor specific algorithms.

10.16.2 Peripheral Device Upstream Port State Machine

The following sections provide a functional description of a state machine that exhibits correct peripheral device behavior for when to connect on SuperSpeed or USB 2.0. Figure 10-25 is an illustration of the peripheral device upstream port state machine.



[1] Peripheral Device must disconnect on USB 2.0 within tUSB2SwitchDisconnect of entering this state.
[2] If USPORT.Powered on was entered from any state except USPORT.Disabled, then this transition shall take place if Far-end Receiver Terminations (RRX-DC) are not detected after eight successive Rx.Detect.Quiet to Rx.Detect.Active transitions. If USPORT.Powered on was entered from the USPORT.Disabled state, then this transition shall take place the first time that Far-End Receiver Terminations are not detected in the Rx.Detect.Active substate.
[3] Disabled count is incremented each time the "Disabled" state is entered. Disabled count is reset to "0" each time the link completes Port Configuration.

U-171

Figure 10-25. Peripheral Upstream Device Port State Machine

10.16.2.1 USDPORT.Powered-off

The USDPORT.Powered-off state is the default state for a peripheral device. A peripheral device shall transition into this state if any of the following situations occur:

- From any state when VBUS is invalid.

In this state, the port's link shall be in the SS.Disabled state and the USB 2.0 pull-up is not applied. The corresponding peripheral USB state shall be Attached.

10.16.2.2 USDPORT.Powered on

A port shall transition into this state if any of the following situations occur:

- From the USDPORT.Powered-off state when VBUS becomes valid (and local power is valid if required).
- From the USDPORT.Error state when the link receives a warm reset or Far-end terminations are removed.
- From the USDPORT.Connected/Enabled state when the link receives a Warm Reset.
- From the USDPORT.Disabled state if the port receives a USB 2.0 reset.
- From the USDPORT.Training state when the link receives a Warm Reset.

In this state, the port's link shall be in the Rx.Detect state. The corresponding peripheral device USB state shall be Powered (Far-end Receiver Termination substate).

If the transition is from the USDPORT.Disabled state the USB 2.0 pull-up shall remain enabled. If the transition is from any other state the USB 2.0 pull-up shall not be enabled.

10.16.2.3 USDPORT.Training

A port transitions to this state from the USDPORT.Powered-on state when SuperSpeed Far-end Receiver Terminations are detected.

In this state, the port's link shall be in the Polling state. The corresponding peripheral device USB state shall be Powered (Link Training substate).

As noted in Figure 10-25, the peripheral device shall disconnect on USB 2.0 within tUSB2SwitchDisconnect after entering this state.

10.16.2.4 USDPORT.Connected/Enabled

A port transitions to this state from the USDPORT.Training state when its link enters U0 from Polling.Idle. A port remains in this state during hot reset. When a hot reset is completed, the corresponding peripheral device USB state shall transition to Default.

In this state, the SuperSpeed link is in U0, U1, U2, U3 or Recovery and the USB 2.0 pull-up is not applied. The corresponding peripheral device USB state shall be Default, Address, or Configured.

10.16.2.5 USDPORT.Error

A port transitions to this state when a serious error condition occurred while attempting to operate the link. A port transitions to this state if the following situation occurs:

- From the USDPORT.Connected/Enabled state if the link enters Recovery and times out without recovering.

In this state, the port's link shall be in the SS.Inactive state. The corresponding peripheral device USB state shall be Error.

A port exits the USDPORT.Error state only if a Warm Reset is received on the link or if Far-end Receiver Terminations are removed.

10.16.2.6 USDPORT.Disabled

A port transitions to this state

- From the USDPORT.Powered on state when Far-end Receiver Terminations are not detected as per the rules described below:

  If USDPORT.Powered on was entered from any state except USDPORT.Disabled then this transition shall take place if Far-end Receiver Terminations (RRX-DC) are not detected after eight successive Rx.Detect.Quiet to Rx.Detect.Active transitions.

  If USDPORT.Powered on was entered from the USDPORT.Disabled state, then this transition shall take place the first time that Far-end Receiver Terminations are not detected in the Rx.Detect.Active substate.
- From the USDPort.Training state if a timeout occurs on any Polling substate (see Figure 7-16).
- From the USDPort.Connected state, if the Port Configuration process times out (see Section 8.4.5).

A count (*Disabled_count*) shall be maintained of each entry into the USDPort.Disabled state.

- Count is initialized to "0" upon power on reset.
- The count is incremented upon each entry into the USDPort.Disabled state (i.e., each increment is the result of one entry into the USDPort.Disabled state).
- If the count equals 3, the port transitions to the Disabled.Error state.
- The count is reset to "0" upon a successful completion of the Port Configuration process.

In this state, the port's link shall be in the SS.Disabled state. The corresponding peripheral device USB state shall be USB 2.0 Device States.

10.16.2.7 USDPORT.Disabled_Error

A port transitions to this state from the USDPort.Disabled state when *Disabled_Count = 3*

- The port shall remain in USDPORT.Disabled_Error state if the port's link receives a USB 2.0 reset.

In this state, a fatal error has been detected on the port's link and the link shall be in the SS.Disabled state. The corresponding peripheral device USB state shall be USB 2.0 Device States.

10.17 Hub Chapter Parameters

Table 10-16 includes a complete list of the parameters used in the hub chapter.

Table 10-16. Hub Parameters

Name	Description	Min	Max	Units
tDownLinkStateChange	Time from receiving the route string of a header packet directed to a downstream port that is in a low power link state to initiating a return to U0 on the downstream link.	0	100	ns
sDataSymbolsBabble	The number of symbols in a data packet payload after the DPPSTART ordered set without a Data Packet Payload ending frame ordered set or DPPABORT ordered set that shall cause a device to detect the packet is invalid.	1030	N/A	symbols
tHubPropRemoteWakeUpstream	Time from start of remote wakeup signaling on the downstream port a hub to when the hub must propagate the remote wakeup signaling on its upstream port if the upstream port link is in U3.	0	1	ms
tHubDriveRemoteWakeDownstream	Time from receiving a SetPortFeature(PORT_LINK_STATE) U0 for a downstream port with a link in U3 to driving remote wakeup signaling on the link.	0	1000	ns
tHubPort2PortExitLat	Time from a downstream port's link initiating a U-state change to when a hub must initiate a U-state change on the upstream port's link (when required).	0	1	µs
aCurrentUnit	Unit for reporting the current draw of hub controller circuitry in the hub descriptor.		4	mA
nMaxHubPorts	Maximum number of ports on a USB 3.0 hub.		15	Ports
tTimeForResetError	If the downstream port link remains in RxDetect.Active or RxDetect.Quiet for this length of time during a warm reset, the reset is considered to have failed.	100	200	ms
tCheckSuperSpeedOnReset	Time from when a device (not a hub) detects a USB 2.0 bus reset to when the device port must enter the USPORT.Powered-On state.	0	1	ms
tUSB2SwitchDisconnect	Time from when a device (not a hub) enters USPORT.Training to when the device must disconnect on the USB 2.0 interface if the device is connected on USB 2.0.	0	1	ms
tPropagationDelayJitterLimit	Variation from the minimum time between when the last symbol of a header packet routed to a downstream port with a link in U0 is received on a hub upstream port and when the first symbol of the header packet is transmitted on the hub downstream port. ITP propagation shall meet tPropagationDelayJitterLimit for all downstream ports that transmit the ITP.	-0	+32	ns
nSkipSymbolLimit	Average number of symbols between transmitted SKP ordered sets.	354	354	Symbols

Name	Description	Min	Max	Units
tHubDelay	This timing defines the maximum delay in nanoseconds a hub can introduce while forwarding packets in either direction. The time is measured from receipt of the last symbol of the packet by the receiving port until the transmitting port sends the first framing of the packet, when both the receiving and transmitting links are in U0 and the following conditions are met: • No Link Commands or SKP ordered sets or other packets are in flight. • Remote Rx Header Buffer Credit Count of the transmitting port is not zero. • Tx Header Buffer of the transmitting port is empty. A hub reports the actual delay via the wHubDelay field in SuperSpeed Hub Descriptor.		400	ns
tDSPortEnabledToU3	Time from when a downstream port enters DSPORT.ENABLED when the upstream hub port is in U3 and remote wakeup is disabled to when the downstream port shall initiate a transition to the U3 link state.	0	1	s

11 Interoperability and Power Delivery

This chapter defines interoperability and power delivery requirements for USB 3.0. Areas covered include USB 3.0 host and device support for USB 2.0 operation, and USB 3.0 VBUS power consumption limits.

Table 11-1 lists the compatibility matrix for USB 3.0 and USB 2.0. The implication of identifying a host port as supporting USB 3.0 is that both hardware and software support for USB 3.0 is in place; otherwise the port shall only be identified as a USB 2.0 port.

Table 11-1. USB 3.0 and USB 2.0 Interoperability

USB Host Port	USB Device Capability	Connected Mode
USB 2.0	USB 2.0	USB 2.0 high-speed, full-speed, or low-speed
	USB 3.0	USB 2.0 high-speed, full-speed, or low-speed
USB 3.0	USB 2.0	USB 2.0 high-speed, full-speed, or low-speed
	USB 3.0	USB 3.0 SuperSpeed

It should be noted that USB 3.0 devices are not required to be backward compatible with USB 1.1 host ports although supporting full-speed and low-speed modes are allowed.

11.1 USB 3.0 Host Support for USB 2.0

USB 3.0-capable ports on hosts shall also support USB 2.0 operation in order to enable backward compatibility with USB 2.0 devices. It should be noted, however, that USB 3.0-capable hosts are not required to support USB 3.0 operation on all of the ports available on the host, i.e., some USB 3.0-capable hosts may have a mix of USB 2.0-only and USB 3.0-capable ports.

To address the situation where a USB 3.0 device is connected to a USB 2.0-only port on a USB 3.0-capable host, after establishing a USB 2.0 high-speed connection with the device, it is recommended that the host inform the user that the device will support SuperSpeed operation if it is moved to a USB 3.0-capable port on the same host. If a USB 3.0 device is connected to a USB 3.0-capable host via a USB 2.0 hub, it is recommended that the host inform the user that the device will support SuperSpeed operation if it is moved to an appropriate host port or if the hub is replaced with a USB 3.0 hub.

When a USB 3.0 hub is connected to a host's USB 3.0-capable port, both USB 3.0 SuperSpeed and USB 2.0 high-speed bus connections shall be allowed to connect and operate in parallel. There is no requirement for a USB 3.0-capable host to support multiple parallel connections to peripheral devices.

The USB 2.0 capabilities of a USB 3.0 host shall be designed to the USB 2.0 specification and shall meet the USB 2.0 compliance requirements.

11.2 USB 3.0 Hub Support for USB 2.0

All ports, both upstream and downstream, on USB 3.0 hubs shall support USB 2.0 operation in order to enable backward compatibility with USB 2.0 devices.

When another USB 3.0 hub is connected in series with a USB 3.0 hub, both SuperSpeed and USB 2.0 high-speed bus connections shall be allowed to connect and operate in parallel. There is no requirement for a USB 3.0 hub to support multiple parallel connections to peripheral devices.

Within a USB 3.0 hub, both the SuperSpeed and USB 2.0 hub devices shall implement in the hub framework a common standardized ContainerID to enable software to identify the physical relationship of the hub devices. The ContainerID descriptor is a part of the BOS descriptor set.

The USB 2.0 capabilities of a USB 3.0 hub shall be designed to the USB 2.0 specification and shall meet the USB 2.0 compliance requirements.

11.3 USB 3.0 Device Support for USB 2.0

In most cases, backward compatible operation at USB 2.0 signaling is supported by USB 3.0 devices in order that higher capability devices are still useful with lesser capable hosts and hubs. For product installations where support for USB 3.0 operation can be independently assured between the device and the host, such as internal devices that are not user accessible, device support for USB 2.0 may not be necessary. USB 3.0 device certification requirements require support for USB 2.0 for all user attached devices.

For any given USB 3.0 peripheral device within a single physical package, only one USB connection mode, either SuperSpeed or a USB 2.0 speed but not both, shall be established for operation with the host.

Peripheral devices may implement in the device framework a common standardized ContainerID to enable software to identify all of the functional components of a specific device and independent of which speed bus it appears on. All devices within a compound device that support ContainerID shall return the same ContainerID.

The USB 2.0 capabilities of a USB 3.0 device shall be designed to the USB 2.0 specification and shall meet the USB 2.0 compliance requirements. Note that a SuperSpeed device operating in one of the USB 2.0 modes must return 0210H in the bcd version field of the device descriptor.

11.4 Power Distribution

This section describes the USB 3.0 power distribution specification. The USB 2.0 power distribution requirements still apply when a USB 3.0 device is operating at high-speed, full-speed, or low-speed. Note that a USB 3.0 peripheral device shall not draw more than 100 mA until it detects far-end Rx terminations in the unconfigured state.

11.4.1 Classes of Devices and Connections

USB 3.0 provides power over two connectors: the Standard-A connector and the Powered-B connector. The following sections focus on the power delivery requirements for the Standard-A connector.

The power source and sink requirements of different device classes can be simplified with the introduction of the concept of a unit load. A unit load for SuperSpeed has been redefined to be 150 mA. The number of unit loads a device can draw is an absolute maximum, not an average over time. A device may be either low-power at one unit load or high-power, consuming up to six unit loads. All devices default to low-power when first powered. The transition to high-power is under software control. It is the responsibility of software to ensure adequate power is available before allowing devices to consume high-power.

The USB supports a range of power sourcing and power consuming agents; these include the following:

- **Root port hubs:** Are directly attached to the USB Host Controller. Hub power is derived from the same source as the Host Controller. Systems that obtain operating power externally, either AC or DC, must be capable of supplying at least six unit loads to each port. Such ports are called high-power ports. Battery-powered systems may supply either one or six unit loads. Ports that can supply only one unit load are termed low-power ports.
- **Self-powered hubs:** Power for the internal functions and downstream facing ports does not come from VBUS. However, the USB interface of the hub may draw up to one unit load from VBUS on its upstream facing port to allow the interface to function when the remainder of the hub is powered down. Hubs that obtain operating power externally (not from VBUS) must supply six unit loads to each port.
- **Low-power bus-powered devices:** All power to these devices comes from VBUS. They may draw no more than one unit load at any time.
- **High-power bus-powered devices:** All power to these devices comes from VBUS. They must draw no more than one unit load upon power-up and may draw up to six unit loads after being configured.
- **Ports may support the USB Charging Specification.**
- **Self-powered devices:** May draw up to one unit load from VBUS to allow the USB interface to function when the remainder of the function is powered down. All other power comes from an external (not from VBUS) source.

No device shall supply (source) current on VBUS at its upstream facing port at any time. From VBUS on its upstream facing port, a device may only draw (sink) current. Devices must also ensure that the maximum operating current drawn by a device is one unit load until configured.

11.4.1.1 Self-powered Hubs

Self-powered hubs have a local power supply that furnishes power to any non-removable functions and to all downstream facing ports, as shown in Figure 11-1. Power for the Hub Controller, however, may be supplied from the upstream VBUS (a "hybrid" powered hub) or the local power supply. The advantage of supplying the Hub Controller from the upstream supply is that communication from the host is possible even if the device's power supply remains off. This makes it possible to differentiate between a disconnected and an unpowered device. If the hub draws power for its upstream facing port from VBUS, it may not draw more than one unit load.

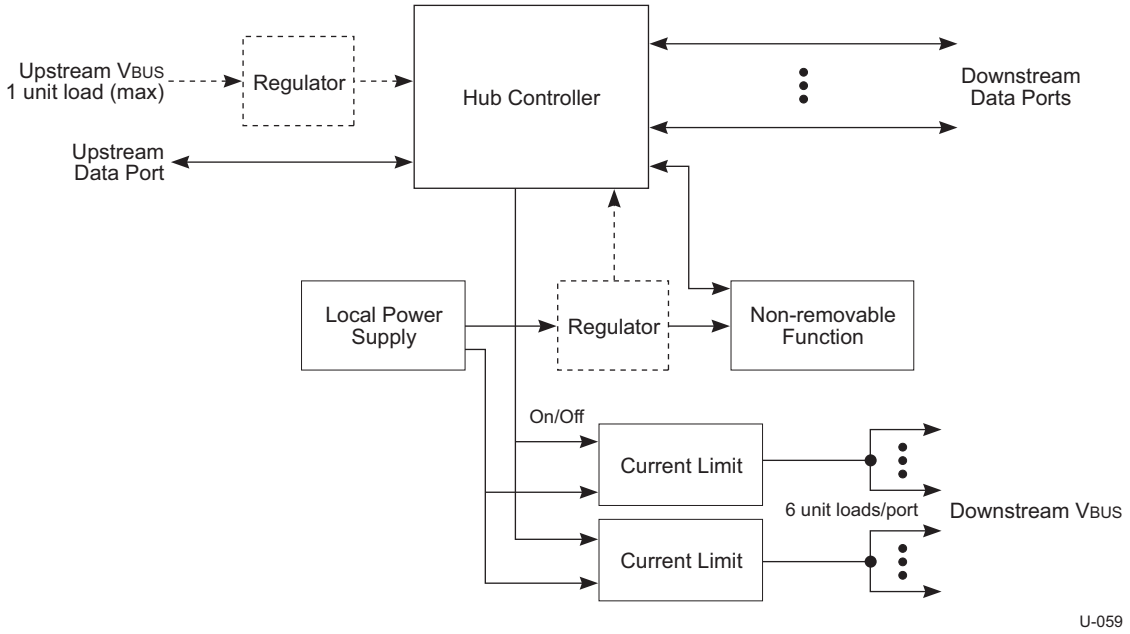


Figure 11-1. Compound Self-powered Hub

The maximum number of ports that can be supported is limited by the capability of the local VBUS supply.

11.4.1.1.1 Over-current Protection

The host and all self-powered hubs must implement over-current protection for safety reasons, and the hub must have a way to detect the over-current condition and report it to the USB software. Should the aggregate current drawn by a gang of downstream facing ports exceed a preset value, the over-current protection circuit removes or reduces power from all affected downstream facing ports. The over-current condition is reported through the hub to the Host Controller, as described in Section 10.11.5. The preset value cannot exceed 5.0 A and must be sufficiently higher than the maximum allowable port current or time delayed such that transient currents (e.g., during power up or dynamic attach or reconfiguration) do not trip the over-current protector. If an over-current condition occurs on any port, subsequent operation of the USB is not guaranteed, and once the condition is removed, it may be necessary to reinitialize the bus as would be done upon power-up. The over-current limiting mechanism must be resettable without user mechanical intervention.

Polymeric PTCs and solid-state switches are examples of methods that can be used for over-current limiting.

11.4.1.2 Low-power Bus-powered Devices

A low-power device is one that draws up to one unit load from the USB cable when operational. Figure 11-2 shows a typical bus-powered, low-power device, such as a mouse. Low-power regulation can be integrated into the device silicon. Low-power devices must be capable of operating with input VBUS voltages as low as 4.00 V, measured at the plug end of the cable.

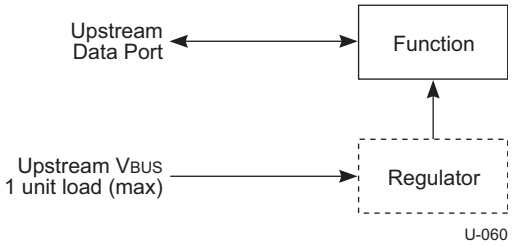


Figure 11-2. Low-power Bus-powered Function

11.4.1.3 High-power Bus-powered Devices

A device is defined as being high-power if, when fully powered, it draws over one but no more than six unit loads from the USB cable. A high-power device requires staged switching of power. It must first come up in a reduced power state of less than one unit load. At bus enumeration time, its total power requirements are obtained and compared against the available power budget. If sufficient power exists, the remainder of the device may be powered on. High-power devices shall be capable of operating with an input voltage as low as 4.00 V. They must also be capable of operating at full power (up to six unit loads) with an input voltage of 4.00 V measured at the device side of the B-series receptacle.

A typical high-power device is shown in Figure 11-3. The device's electronics have been partitioned into two sections. The device controller contains the minimum amount of circuitry necessary to permit enumeration and power budgeting. The remainder of the device resides in the function block.

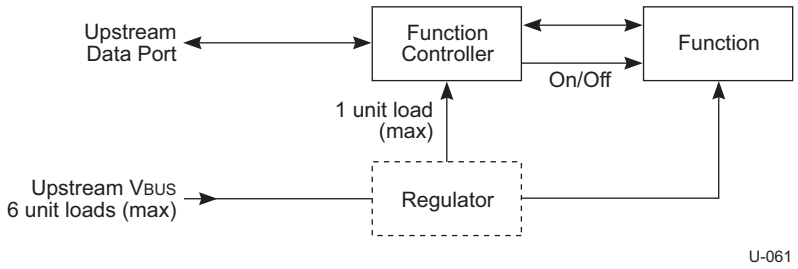


Figure 11-3. High-power Bus-powered Function

11.4.1.4 Self-powered Devices

Figure 11-4 shows a typical self-powered device. The device controller is powered either from the upstream bus via a low-power regulator or from the local power supply. The advantage of the former scheme is that it permits detection and enumeration of a self-powered device whose local power supply is turned off. The maximum upstream power that the device controller can draw is one unit load, and the regulator block must implement inrush current limiting. The amount of power that the device block may draw is limited only by the local power supply. Because the local power supply is not required to power any downstream bus ports, it does not need to implement current limiting, soft start, or power switching.

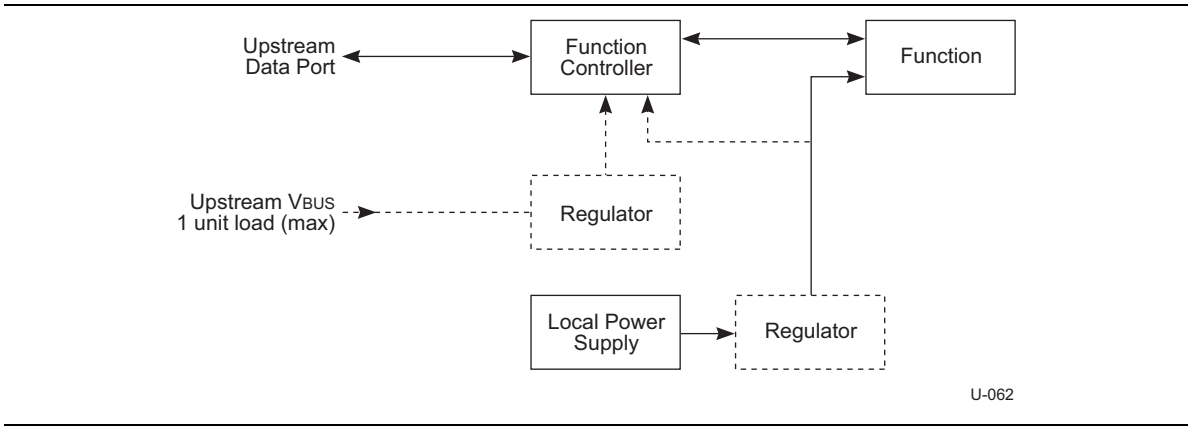


Figure 11-4. Self-powered Function

11.4.2 Steady-State Voltage Drop Budget

The steady–state voltage drop budget is derived from the following assumptions:

- The nominal 5 V ± 5% source (host or hub) is 4.75 V to 5.25 V.
- The voltage supplied at the connector of hub or root ports shall be between 4.45 V to 5.25 V.
- The maximum voltage drop (for detachable cables) between the A-series plug and B-series plug on VBUS is 171 mV.
- The maximum current for the calculations is 0.9 A.
- The maximum voltage drop for all cables between upstream and downstream on GND is 171 mV.
- The maximum voltage drop for all mated connectors is 27 mV.
- All hubs and peripheral devices shall be able to provide configuration information with as little as 4.00 V at the device end of their B-series receptacle. Both low and high-power devices need to be operational with this minimum voltage.

Figure 11-5 shows the minimum allowable voltages. Note that under transient conditions, the supply at the device can drop to 3.67 V for a brief moment.

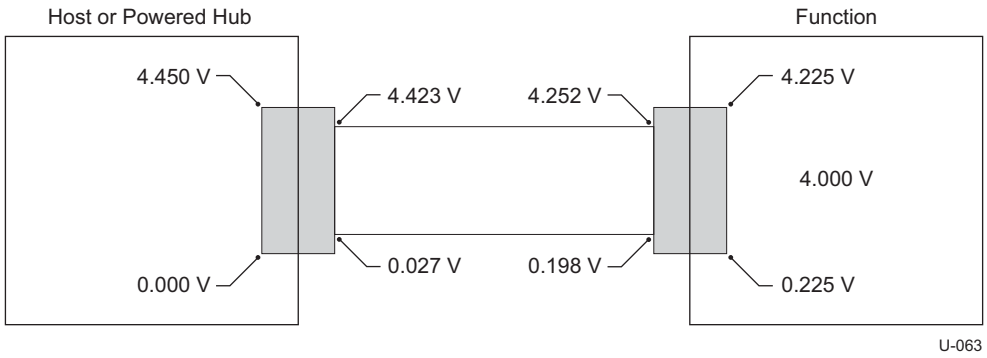


Figure 11-5. Worst-case Voltage Drop Topology (Steady State)

Note: the following assumptions were used in Figure 11-5:

- 3 meter cable assembly with A-series and B-series plugs
- #22AWG wire used for power and ground (0.019 Ω/foot)
- A-series and B-series plug/receptacle pair have a contact resistance of 30 mΩ
- Wire ~380 mΩ series resistance
- IR Drop at device = (((2 * 30 mΩ) + 190 mΩ) * 900 mA) * 2 or 0.450 V

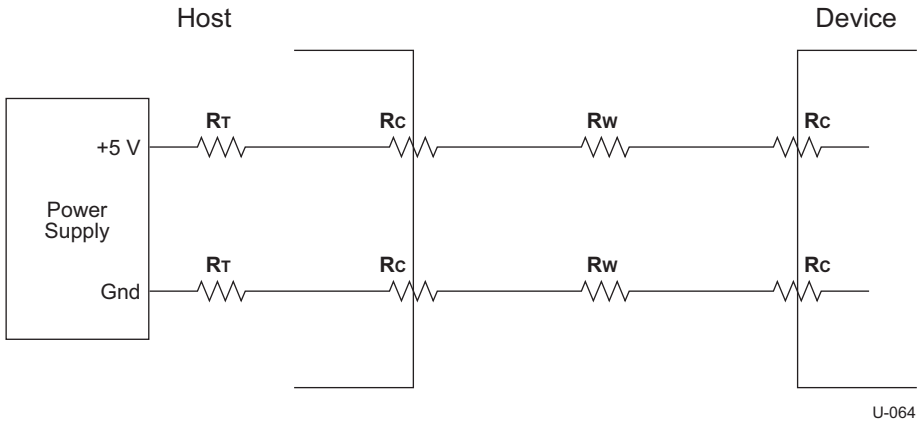


Figure 11-6. Worst-case Voltage Drop Analysis Using Equivalent Resistance

Rt	Host Trace Resistance:	0.167 Ω
Rc	Mated Connector Resistance:	0.030 Ω
Rw	Cable Resistance:	0.190 Ω

11.4.3 Power Control During Suspend/Resume

All USB devices may draw up to 2.5 mA during suspend. When configured, bus-powered compound devices may consume a suspend current of up to 12.5 mA. This 12.5 mA budget includes 2.5 mA suspend current for the internal hub plus 2.5 mA suspend current for each port on that internal hub having attached internal functions, up to a maximum of four ports. When computing suspend current, the current from VBUS through the bus pull-up and pull-down resistors must be included.

While in the Suspend state, a device may briefly draw more than the average current. The amplitude of the current spike cannot exceed the device power allocation 150 mA (or 900 mA). A maximum of 1.0 second is allowed for an averaging interval. The average current cannot exceed the average suspend current limit (ICCS, see Table 11-2) during any 1.0-second interval. The profile of the current spike is restricted so the transient response of the power supply (which may be an efficient, low-capacity, trickle power supply) is not overwhelmed. The rising edge of the current spike must be no more than 100 mA/µs. Downstream facing ports must be able to absorb the 900 mA peak current spike and meet the voltage droop requirements defined for inrush current during dynamic attach. Figure 11-7 illustrates a typical example profile for an averaging interval.

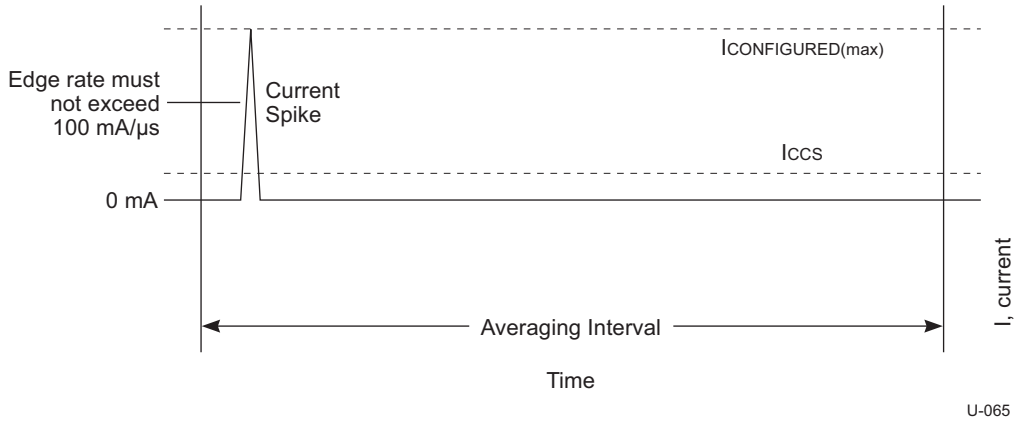


Figure 11-7. Typical Suspend Current Averaging Profile

Devices are responsible for handling the bus voltage reduction due to the inductive and resistive effects of the cable. When a hub is in the Suspend state, it must still be able to provide the maximum current per port (six unit loads per port for self-powered hubs). This is necessary to support remote wakeup-capable devices that will power-up while the remainder of the system is still suspended. Such devices, when enabled to do remote wakeup, must drive resume signaling upstream within 10 ms of starting to draw the higher, non-suspend current. Devices not capable of remote wakeup must not draw the higher current when suspended.

When devices wakeup, either by themselves (remote wakeup) or by seeing resume signaling, they must limit the inrush current on VBUS. The device must have sufficient on-board bypass capacitance or a controlled power-on sequence such that the current drawn from the hub does not exceed the maximum current capability of the port at any time while the device is waking up.

11.4.4 Dynamic Attach and Detach

The act of plugging or unplugging a hub or peripheral device must not affect the functionality of another device on other segments of the network. Unplugging a device will stop any transactions in progress between that device and the host. However, the hub or root port to which this device was attached will recover from this condition and will alert the host that the port has been disconnected.

11.4.4.1 Inrush Current Limiting

When a peripheral device or hub is plugged into the network, it has a certain amount of on-board capacitance between VBUS and ground. In addition, the regulator on the device may supply current to its output bypass capacitance and to the device as soon as power is applied. Consequently, if no measures are taken to prevent it, there could be a surge of current into the device which might pull the VBUS on the hub below its minimum operating level. Inrush currents can also occur when a high-power device is switched into its high-power mode. This problem must be solved by limiting the inrush current and by providing sufficient capacitance in each hub to prevent the power supplied to the other ports from going out of tolerance. An additional motivation for limiting inrush current is to minimize contact arcing, thereby prolonging connector contact life.

The maximum droop possible in the hub VBUS is 330 mV. In order to meet this requirement, the following conditions must be met:

- The maximum load (C_{RPB}) that can be placed at the downstream end of a cable is 10 μF in parallel with as small as a 27 Ω resistance. The 10 μF capacitance represents any bypass capacitor directly connected across the VBUS lines in the device plus any capacitive effects visible through the regulator in the device. The 27 Ω resistance represents one unit load of current drawn by the device during connect.
- If more bypass capacitance is required in the device, then the device must incorporate some form of VBUS surge current limiting, such that it matches the characteristics of the above load.
- The hub downstream facing port VBUS power lines must be bypassed (C_{HPB}) with no less than 120 μF of low-ESR capacitance per hub. Standard bypass methods should be used to minimize inductance and resistance between the bypass capacitors and the connectors to reduce droop. The bypass capacitors themselves should have a low dissipation factor to allow decoupling at higher frequencies.

The upstream facing port of a hub is also required to meet the above requirements.

A high-power bus-powered device that is switching from a lower power configuration to a higher power configuration must not cause droop >330 mV on the VBUS at its upstream hub. The device can meet this by ensuring that changes in the capacitive load it presents do not exceed 10 μF.

11.4.4.2 Dynamic Detach

When a device is detached from the network with power flowing in the cable, the inductance of the cable will cause a large flyback voltage to occur on the open end of the device cable. This flyback voltage is not destructive. Proper bypass measures on the hub ports will suppress any coupled noise. This will require some low capacitance, very low inductance bypass capacitors on each hub port connector. The flyback voltage and the noise it creates are also moderated by the bypass capacitance on the device end of the cable. Also, there must be some minimum capacitance on the device end of the cable to ensure that the inductive flyback on the open end of the cable does not cause the voltage on the device end to reverse polarity. A minimum of 1.0 μF is recommended for bypass across V_{BUS}.

11.4.5 V_{BUS} Electrical Characteristics

Table 11-2. DC Electrical Characteristics

Parameter	Symbol	Conditions	Min.	Max.	Units
Supply Voltage:					
Port (downstream connector)	V_{BUS}		4.45	5.25	V
Port (upstream connector)	V_{BUS}		4.0		V
Supply Current:					
High-power Hub Port (out)	I_{CCPRT}		900		mA
Low-power Hub Port (out)	I_{CCUPT}		150		mA
High-power Peripheral Device (in)	I_{CCHPF}			900	mA
Low-power Peripheral Device (in)	I_{CCLPF}			150	mA
Unconfigured Device (in)	I_{CCINIT}			150	mA
Suspended High-power Device	I_{CCS}			2.5	mA

11.4.6 Powered-B Connector

The Powered-B connector was defined to allow devices such as printers to connect to and power devices such as Wireless USB adapters. This improves usability by eliminating the need for an external power supply (e.g., “wall-wart”) to power the adapter.

The Powered-B receptacle shall meet the following electrical requirements:

- The Powered-B receptacle shall supply 5 V ± 10% over the entire load range (0-1 A).
- The Powered-B receptacle shall be capable of supplying 1 A.
- The Powered-B receptacle shall implement over-current protection for safety reasons.
- The Powered-B receptacle shall deliver full power whenever the device is turned on or in standby.
- A device that exposes a Powered-B receptacle shall operate as a low-power device.
- A device that is powered (in whole or in part) by a Powered-B receptacle shall not expose any Standard-A ports

Note: Power distribution defined in this chapter applies to the devices with the Powered-B connector. The cable assembly has to deliver VBUS to the device's USB interface via the VPWR and GND pins of the Powered-B connector.

11.4.7 Wire Gauge Table

Table 11-3 is a table of VBUS/Gnd wire gauges showing the relationship between gauge and maximum length in order to achieve the previously cited voltage drop values. The user should note that these lengths are the maximum length possible to meet the voltage drop budget, thus gauges smaller and lengths greater than the table values will fail to deliver the expected voltage value.

Table 11-3. VBUS/Gnd Wire Gauge vs. Maximum Length

American Stranded Wire Gauge (AWG) on VBUS/Gnd	Ohms Per 100 Meters (Maximum)	Maximum Cable Length (Meters)
28	23.20	0.8
26	14.60	1.3
24	9.09	2.0
22	5.74	3.0
20	3.58	5.3

A Symbol Encoding

Table A-1 shows the byte-to-Symbol encodings for data characters. Table A-2 shows the Symbol encodings for the Special Symbols. RD- and RD+ refer to the Running Disparity of the Symbol sequence on a per-Lane basis.

Table A-1. 8b/10b Data Symbol Codes

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj (binary)	Current RD+ abcdei fghj (binary)
D0.0	00	000 00000	100111 0100	011000 1011
D1.0	01	000 00001	011101 0100	100010 1011
D2.0	02	000 00010	101101 0100	010010 1011
D3.0	03	000 00011	110001 1011	110001 0100
D4.0	04	000 00100	110101 0100	001010 1011
D5.0	05	000 00101	101001 1011	101001 0100
D6.0	06	000 00110	011001 1011	011001 0100
D7.0	07	000 00111	111000 1011	000111 0100
D8.0	08	000 01000	111001 0100	000110 1011
D9.0	09	000 01001	100101 1011	100101 0100
D10.0	0A	000 01010	010101 1011	010101 0100
D11.0	0B	000 01011	110100 1011	110100 0100
D12.0	0C	000 01100	001101 1011	001101 0100
D13.0	0D	000 01101	101100 1011	101100 0100
D14.0	0E	000 01110	011100 1011	011100 0100
D15.0	0F	000 01111	010111 0100	101000 1011
D16.0	10	000 10000	011011 0100	100100 1011
D17.0	11	000 10001	100011 1011	100011 0100
D18.0	12	000 10010	010011 1011	010011 0100
D19.0	13	000 10011	110010 1011	110010 0100
D20.0	14	000 10100	001011 1011	001011 0100
D21.0	15	000 10101	101010 1011	101010 0100
D22.0	16	000 10110	011010 1011	011010 0100
D23.0	17	000 10111	111010 0100	000101 1011
D24.0	18	000 11000	110011 0100	001100 1011
D25.0	19	000 11001	100110 1011	100110 0100
D26.0	1A	000 11010	010110 1011	010110 0100
D27.0	1B	000 11011	110110 0100	001001 1011
D28.0	1C	000 11100	001110 1011	001110 0100
D29.0	1D	000 11101	101110 0100	010001 1011
D30.0	1E	000 11110	011110 0100	100001 1011
D31.0	1F	000 11111	101011 0100	010100 1011

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj (binary)	Current RD+ abcdei fghj (binary)
D0.1	20	001 00000	100111 1001	011000 1001
D1.1	21	001 00001	011101 1001	100010 1001
D2.1	22	001 00010	101101 1001	010010 1001
D3.1	23	001 00011	110001 1001	110001 1001
D4.1	24	001 00100	110101 1001	001010 1001
D5.1	25	001 00101	101001 1001	101001 1001
D6.1	26	001 00110	011001 1001	011001 1001
D7.1	27	001 00111	111000 1001	000111 1001
D8.1	28	001 01000	111001 1001	000110 1001
D9.1	29	001 01001	100101 1001	100101 1001
D10.1	2A	001 01010	010101 1001	010101 1001
D11.1	2B	001 01011	110100 1001	110100 1001
D12.1	2C	001 01100	001101 1001	001101 1001
D13.1	2D	001 01101	101100 1001	101100 1001
D14.1	2E	001 01110	011100 1001	011100 1001
D15.1	2F	001 01111	010111 1001	101000 1001
D16.1	30	001 10000	011011 1001	100100 1001
D17.1	31	001 10001	100011 1001	100011 1001
D18.1	32	001 10010	010011 1001	010011 1001
D19.1	33	001 10011	110010 1001	110010 1001
D20.1	34	001 10100	001011 1001	001011 1001
D21.1	35	001 10101	101010 1001	101010 1001
D22.1	36	001 10110	011010 1001	011010 1001
D23.1	37	001 10111	111010 1001	000101 1001
D24.1	38	001 11000	110011 1001	001100 1001
D25.1	39	001 11001	100110 1001	100110 1001
D26.1	3A	001 11010	010110 1001	010110 1001
D27.1	3B	001 11011	110110 1001	001001 1001
D28.1	3C	001 11100	001110 1001	001110 1001
D29.1	3D	001 11101	101110 1001	010001 1001
D30.1	3E	001 11110	011110 1001	100001 1001
D31.1	3F	001 11111	101011 1001	010100 1001
D0.2	40	010 00000	100111 0101	011000 0101
D1.2	41	010 00001	011101 0101	100010 0101
D2.2	42	010 00010	101101 0101	010010 0101
D3.2	43	010 00011	110001 0101	110001 0101
D4.2	44	010 00100	110101 0101	001010 0101
D5.2	45	010 00101	101001 0101	101001 0101
D6.2	46	010 00110	011001 0101	011001 0101
D7.2	47	010 00111	111000 0101	000111 0101

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj (binary)	Current RD+ abcdei fghj (binary)
D8.2	48	010 01000	111001 0101	000110 0101
D9.2	49	010 01001	100101 0101	100101 0101
D10.2	4A	010 01010	010101 0101	010101 0101
D11.2	4B	010 01011	110100 0101	110100 0101
D12.2	4C	010 01100	001101 0101	001101 0101
D13.2	4D	010 01101	101100 0101	101100 0101
D14.2	4E	010 01110	011100 0101	011100 0101
D15.2	4F	010 01111	010111 0101	101000 0101
D16.2	50	010 10000	011011 0101	100100 0101
D17.2	51	010 10001	100011 0101	100011 0101
D18.2	52	010 10010	010011 0101	010011 0101
D19.2	53	010 10011	110010 0101	110010 0101
D20.2	54	010 10100	001011 0101	001011 0101
D21.2	55	010 10101	101010 0101	101010 0101
D22.2	56	010 10110	011010 0101	011010 0101
D23.2	57	010 10111	111010 0101	000101 0101
D24.2	58	010 11000	110011 0101	001100 0101
D25.2	59	010 11001	100110 0101	100110 0101
D26.2	5A	010 11010	010110 0101	010110 0101
D27.2	5B	010 11011	110110 0101	001001 0101
D28.2	5C	010 11100	001110 0101	001110 0101
D29.2	5D	010 11101	101110 0101	010001 0101
D30.2	5E	010 11110	011110 0101	100001 0101
D31.2	5F	010 11111	101011 0101	010100 0101
D0.3	60	011 00000	100111 0011	011000 1100
D1.3	61	011 00001	011101 0011	100010 1100
D2.3	62	011 00010	101101 0011	010010 1100
D3.3	63	011 00011	110001 1100	110001 0011
D4.3	64	011 00100	110101 0011	001010 1100
D5.3	65	011 00101	101001 1100	101001 0011
D6.3	66	011 00110	011001 1100	011001 0011
D7.3	67	011 00111	111000 1100	000111 0011
D8.3	68	011 01000	111001 0011	000110 1100
D9.3	69	011 01001	100101 1100	100101 0011
D10.3	6A	011 01010	010101 1100	010101 0011
D11.3	6B	011 01011	110100 1100	110100 0011
D12.3	6C	011 01100	001101 1100	001101 0011
D13.3	6D	011 01101	101100 1100	101100 0011
D14.3	6E	011 01110	011100 1100	011100 0011
D15.3	6F	011 01111	010111 0011	101000 1100

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj (binary)	Current RD+ abcdei fghj (binary)
D16.3	70	011 10000	011011 0011	100100 1100
D17.3	71	011 10001	100011 1100	100011 0011
D18.3	72	011 10010	010011 1100	010011 0011
D19.3	73	011 10011	110010 1100	110010 0011
D20.3	74	011 10100	001011 1100	001011 0011
D21.3	75	011 10101	101010 1100	101010 0011
D22.3	76	011 10110	011010 1100	011010 0011
D23.3	77	011 10111	111010 0011	000101 1100
D24.3	78	011 11000	110011 0011	001100 1100
D25.3	79	011 11001	100110 1100	100110 0011
D26.3	7A	011 11010	010110 1100	010110 0011
D27.3	7B	011 11011	110110 0011	001001 1100
D28.3	7C	011 11100	001110 1100	001110 0011
D29.3	7D	011 11101	101110 0011	010001 1100
D30.3	7E	011 11110	011110 0011	100001 1100
D31.3	7F	011 11111	101011 0011	010100 1100
D0.4	80	100 00000	100111 0010	011000 1101
D1.4	81	100 00001	011101 0010	100010 1101
D2.4	82	100 00010	101101 0010	010010 1101
D3.4	83	100 00011	110001 1101	110001 0010
D4.4	84	100 00100	110101 0010	001010 1101
D5.4	85	100 00101	101001 1101	101001 0010
D6.4	86	100 00110	011001 1101	011001 0010
D7.4	87	100 00111	111000 1101	000111 0010
D8.4	88	100 01000	111001 0010	000110 1101
D9.4	89	100 01001	100101 1101	100101 0010
D10.4	8A	100 01010	010101 1101	010101 0010
D11.4	8B	100 01011	110100 1101	110100 0010
D12.4	8C	100 01100	001101 1101	001101 0010
D13.4	8D	100 01101	101100 1101	101100 0010
D14.4	8E	100 01110	011100 1101	011100 0010
D15.4	8F	100 01111	010111 0010	101000 1101
D16.4	90	100 10000	011011 0010	100100 1101
D17.4	91	100 10001	100011 1101	100011 0010
D18.4	92	100 10010	010011 1101	010011 0010
D19.4	93	100 10011	110010 1101	110010 0010
D20.4	94	100 10100	001011 1101	001011 0010
D21.4	95	100 10101	101010 1101	101010 0010
D22.4	96	100 10110	011010 1101	011010 0010
D23.4	97	100 10111	111010 0010	000101 1101

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj (binary)	Current RD+ abcdei fghj (binary)
D24.4	98	100 11000	110011 0010	001100 1101
D25.4	99	100 11001	100110 1101	100110 0010
D26.4	9A	100 11010	010110 1101	010110 0010
D27.4	9B	100 11011	110110 0010	001001 1101
D28.4	9C	100 11100	001110 1101	001110 0010
D29.4	9D	100 11101	101110 0010	010001 1101
D30.4	9E	100 11110	011110 0010	100001 1101
D31.4	9F	100 11111	101011 0010	010100 1101
D0.5	A0	101 00000	100111 1010	011000 1010
D1.5	A1	101 00001	011101 1010	100010 1010
D2.5	A2	101 00010	101101 1010	010010 1010
D3.5	A3	101 00011	110001 1010	110001 1010
D4.5	A4	101 00100	110101 1010	001010 1010
D5.5	A5	101 00101	101001 1010	101001 1010
D6.5	A6	101 00110	011001 1010	011001 1010
D7.5	A7	101 00111	111000 1010	000111 1010
D8.5	A8	101 01000	111001 1010	000110 1010
D9.5	A9	101 01001	100101 1010	100101 1010
D10.5	AA	101 01010	010101 1010	010101 1010
D11.5	AB	101 01011	110100 1010	110100 1010
D12.5	AC	101 01100	001101 1010	001101 1010
D13.5	AD	101 01101	101100 1010	101100 1010
D14.5	AE	101 01110	011100 1010	011100 1010
D15.5	AF	101 01111	010111 1010	101000 1010
D16.5	B0	101 10000	011011 1010	100100 1010
D17.5	B1	101 10001	100011 1010	100011 1010
D18.5	B2	101 10010	010011 1010	010011 1010
D19.5	B3	101 10011	110010 1010	110010 1010
D20.5	B4	101 10100	001011 1010	001011 1010
D21.5	B5	101 10101	101010 1010	101010 1010
D22.5	B6	101 10110	011010 1010	011010 1010
D23.5	B7	101 10111	111010 1010	000101 1010
D24.5	B8	101 11000	110011 1010	001100 1010
D25.5	B9	101 11001	100110 1010	100110 1010
D26.5	BA	101 11010	010110 1010	010110 1010
D27.5	BB	101 11011	110110 1010	001001 1010
D28.5	BC	101 11100	001110 1010	001110 1010
D29.5	BD	101 11101	101110 1010	010001 1010
D30.5	BE	101 11110	011110 1010	100001 1010
D31.5	BF	101 11111	101011 1010	010100 1010

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj (binary)	Current RD+ abcdei fghj (binary)
D0.6	C0	110 00000	100111 0110	011000 0110
D1.6	C1	110 00001	011101 0110	100010 0110
D2.6	C2	110 00010	101101 0110	010010 0110
D3.6	C3	110 00011	110001 0110	110001 0110
D4.6	C4	110 00100	110101 0110	001010 0110
D5.6	C5	110 00101	101001 0110	101001 0110
D6.6	C6	110 00110	011001 0110	011001 0110
D7.6	C7	110 00111	111000 0110	000111 0110
D8.6	C8	110 01000	111001 0110	000110 0110
D9.6	C9	110 01001	100101 0110	100101 0110
D10.6	CA	110 01010	010101 0110	010101 0110
D11.6	CB	110 01011	110100 0110	110100 0110
D12.6	CC	110 01100	001101 0110	001101 0110
D13.6	CD	110 01101	101100 0110	101100 0110
D14.6	CE	110 01110	011100 0110	011100 0110
D15.6	CF	110 01111	010111 0110	101000 0110
D16.6	D0	110 10000	011011 0110	100100 0110
D17.6	D1	110 10001	100011 0110	100011 0110
D18.6	D2	110 10010	010011 0110	010011 0110
D19.6	D3	110 10011	110010 0110	110010 0110
D20.6	D4	110 10100	001011 0110	001011 0110
D21.6	D5	110 10101	101010 0110	101010 0110
D22.6	D6	110 10110	011010 0110	011010 0110
D23.6	D7	110 10111	111010 0110	000101 0110
D24.6	D8	110 11000	110011 0110	001100 0110
D25.6	D9	110 11001	100110 0110	100110 0110
D26.6	DA	110 11010	010110 0110	010110 0110
D27.6	DB	110 11011	110110 0110	001001 0110
D28.6	DC	110 11100	001110 0110	001110 0110
D29.6	DD	110 11101	101110 0110	010001 0110
D30.6	DE	110 11110	011110 0110	100001 0110
D31.6	DF	110 11111	101011 0110	010100 0110
D0.7	E0	111 00000	100111 0001	011000 1110
D1.7	E1	111 00001	011101 0001	100010 1110
D2.7	E2	111 00010	101101 0001	010010 1110
D3.7	E3	111 00011	110001 1110	110001 0001
D4.7	E4	111 00100	110101 0001	001010 1110
D5.7	E5	111 00101	101001 1110	101001 0001
D6.7	E6	111 00110	011001 1110	011001 0001
D7.7	E7	111 00111	111000 1110	000111 0001

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj (binary)	Current RD+ abcdei fghj (binary)
D8.7	E8	111 01000	111001 0001	000110 1110
D9.7	E9	111 01001	100101 1110	100101 0001
D10.7	EA	111 01010	010101 1110	010101 0001
D11.7	EB	111 01011	110100 1110	110100 1000
D12.7	EC	111 01100	001101 1110	001101 0001
D13.7	ED	111 01101	101100 1110	101100 1000
D14.7	EE	111 01110	011100 1110	011100 1000
D15.7	EF	111 01111	010111 0001	101000 1110
D16.7	F0	111 10000	011011 0001	100100 1110
D17.7	F1	111 10001	100011 0111	100011 0001
D18.7	F2	111 10010	010011 0111	010011 0001
D19.7	F3	111 10011	110010 1110	110010 0001
D20.7	F4	111 10100	001011 0111	001011 0001
D21.7	F5	111 10101	101010 1110	101010 0001
D22.7	F6	111 10110	011010 1110	011010 0001
D23.7	F7	111 10111	111010 0001	000101 1110
D24.7	F8	111 11000	110011 0001	001100 1110
D25.7	F9	111 11001	100110 1110	100110 0001
D26.7	FA	111 11010	010110 1110	010110 0001
D27.7	FB	111 11011	110110 0001	001001 1110
D28.7	FC	111 11100	001110 1110	001110 0001
D29.7	FD	111 11101	101110 0001	010001 1110
D30.7	FE	111 11110	011110 0001	100001 1110
D31.7	FF	111 11111	101011 0001	010100 1110

Table A-2. 8b/10b Special Character Symbol Codes

Data Byte Name	Data Byte Value	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD + abcdei fghj
K28.0	1C	000 11100	001111 0100	110000 1011
K28.1	3C	001 11100	001111 1001	110000 0110
K28.2	5C	010 11100	001111 0101	110000 1010
K28.3	7C	011 11100	001111 0011	110000 1100
K28.4	9C	100 11100	001111 0010	110000 1101
K28.5	BC	101 11100	001111 1010	110000 0101
K28.6	DC	110 11100	001111 0110	110000 1001
K28.7	FC	111 11100	001111 1000	110000 0111
K23.7	F7	111 10111	111010 1000	000101 0111
K27.7	FB	111 11011	110110 1000	001001 0111
K29.7	FD	111 11101	101110 1000	010001 0111
K30.7	FE	111 11110	011110 1000	100001 0111

Note: Only a small fraction of the possible K-characters are defined in this table. Any K-character that decodes to a value that is not in Table A-2 shall be returned as Decode_Error_Substitution (K28.4). Refer to Section 6.3.4 and Table 6-1 for more information.

B Symbol Scrambling

B.1 Data Scrambling

The following subroutines encode and decode an 8-bit value contained in “inbyte” with the LFSR. This is presented as one example only; there are many ways to obtain the proper output. This example demonstrates how to advance the LFSR eight times in one operation and how to XOR the data in one operation. Many other implementations are possible but they must all produce the same output as that shown here.

The following algorithm uses the “C” programming language conventions, where “<<” and “>>” represent the shift left and shift right operators, “>” is the compare greater than operator, and “^” is the exclusive or operator, and “&” is the logical “AND” operator.

```
/*
   this routine implements the serial descrambling algorithm in parallel form
   for the LSFR polynomial:  x^16+x^5+x^4+x^3+1
   this advances the LSFR 8 bits every time it is called
   this requires fewer than 25 xor gates to implement (with a static register)

   The XOR required to advance 8 bits/clock is:
bit   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15

      8   9  10  11  12  13  14  15   0   1   2   3   4   5   6   7
                  8   9  10  11  12  13  14  15
                      8   9  10  11  12  13  14  15
                          8   9  10  11  12  13  14  15

   The serial data is just the reverse of the upper byte:
bit   0   1   2   3   4   5   6   7
     15  14  13  12  11  10   9   8
*/
```

```
int scramble_byte(int inbyte)
{

static int scrambit[16];
static int bit[16];
static int bit_out[16];
static unsigned short lfsr = 0xffff; // 16 bit short for polynomial
int i, outbyte;

    if (inbyte == COMMA)   // if this is a comma
    {
        lfsr = 0xffff;     // reset the LFSR
        return (COMMA);    // and return the same data
    }

    if (inbyte == SKIP)    // don't advance or encode on skip
        return (SKIP);

    for (i=0; i<16;i++)    // convert LFSR to bit array for legibility
        bit[i] = (lfsr >> i) & 1;

    for (i=0; i<8; i++)    // convert byte to be scrambled for legibility
        scrambit[i] = (inbyte >> i) & 1;

    // apply the xor to the data
    if (! (inbyte & 0x100) &&    // if not a KCODE, scramble the data
        ! (TrainingSequence == TRUE))  // and if not in the middle of
    {                                  // a training sequence
        scrambit[0] ^= bit[15];
        scrambit[1] ^= bit[14];
        scrambit[2] ^= bit[13];
        scrambit[3] ^= bit[12];
        scrambit[4] ^= bit[11];
        scrambit[5] ^= bit[10];
        scrambit[6] ^= bit[9];
        scrambit[7] ^= bit[8];
    }
```

```
    // Now advance the LFSR 8 serial clocks
    bit_out[ 0] = bit[ 8];
    bit_out[ 1] = bit[ 9];
    bit_out[ 2] = bit[10];
    bit_out[ 3] = bit[11] ^ bit[ 8];
    bit_out[ 4] = bit[12] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 5] = bit[13] ^ bit[10] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 6] = bit[14] ^ bit[11] ^ bit[10] ^ bit[ 9];
    bit_out[ 7] = bit[15] ^ bit[12] ^ bit[11] ^ bit[10];
    bit_out[ 8] = bit[ 0] ^ bit[13] ^ bit[12] ^ bit[11];
    bit_out[ 9] = bit[ 1] ^ bit[14] ^ bit[13] ^ bit[12];
    bit_out[10] = bit[ 2] ^ bit[15] ^ bit[14] ^ bit[13];
    bit_out[11] = bit[ 3]           ^ bit[15] ^ bit[14];
    bit_out[12] = bit[ 4]                     ^ bit[15];
    bit_out[13] = bit[ 5];
    bit_out[14] = bit[ 6];
    bit_out[15] = bit[ 7];
    lfsr = 0;
    for (i=0; i <16; i++) // convert the LFSR back to an integer
        lfsr += (bit_out[i] << i);

    outbyte = 0;
    for (i= 0; i<8; i++) // convert data back to an integer
        outbyte += (scrambit[i] << i);

    return outbyte;
}

/*  NOTE THAT THE DESCRAMBLE ROUTINE IS IDENTICAL TO THE SCRAMBLE ROUTINE
    this routine implements the serial descrambling algorithm in parallel form
    this advances the lfsr 8 bits every time it is called
    this uses fewer than 25 xor gates to implement (with a static register)
    The XOR tree is the same as the scrambling routine
*/
```

```
int unscramble_byte(int inbyte)
{

    static int descrambit[8];
    static int bit[16];
    static int bit_out[16];
    static unsigned short lfsr = 0xffff;  // 16 bit short for polynomial
    int outbyte, i;

    if (inbyte == COMMA)   // if this is a comma
    {
       lfsr = 0xffff;      // reset the LFSR
       return (COMMA);     // and return the same data
    }

    if (inbyte == SKIP)   // don't advance or encode on skip
        return (SKIP);

    for (i=0; i<16;i++)   // convert the LFSR to bit array for legibility
        bit[i] = (lfsr >> i) & 1;

    for (i=0; i<8; i++)   // convert byte to be de-scrambled for legibility
        descrambit[i] = (inbyte >> i) & 1;

    // apply the xor to the data
    if (! (inbyte & 0x100) &&    // if not a KCODE, scramble the data
        ! (TrainingSequence == TRUE))  // and if not in the middle of
    {                                  // a training sequence
        descrambit[0] ^= bit[15];
        descrambit[1] ^= bit[14];
        descrambit[2] ^= bit[13];
        descrambit[3] ^= bit[12];
        descrambit[4] ^= bit[11];
        descrambit[5] ^= bit[10];
        descrambit[6] ^= bit[9];
        descrambit[7] ^= bit[8];
    }
```

```
    // Now advance the LFSR 8 serial clocks
    bit_out[ 0] = bit[ 8];
    bit_out[ 1] = bit[ 9];
    bit_out[ 2] = bit[10];
    bit_out[ 3] = bit[11] ^ bit[ 8];
    bit_out[ 4] = bit[12] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 5] = bit[13] ^ bit[10] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 6] = bit[14] ^ bit[11] ^ bit[10] ^ bit[ 9];
    bit_out[ 7] = bit[15] ^ bit[12] ^ bit[11] ^ bit[10];
    bit_out[ 8] = bit[ 0] ^ bit[13] ^ bit[12] ^ bit[11];
    bit_out[ 9] = bit[ 1] ^ bit[14] ^ bit[13] ^ bit[12];
    bit_out[10] = bit[ 2] ^ bit[15] ^ bit[14] ^ bit[13];
    bit_out[11] = bit[ 3]           ^ bit[15] ^ bit[14];
    bit_out[12] = bit[ 4]                     ^ bit[15];
    bit_out[13] = bit[ 5];
    bit_out[14] = bit[ 6];
    bit_out[15] = bit[ 7];
    lfsr = 0;
    for (i=0; i <16; i++) // convert the LFSR back to an integer
        lfsr += (bit_out[i] << i);

    outbyte = 0;
    for (i= 0; i<8; i++) // convert data back to an integer
        outbyte += (descrambit[i] << i);

    return outbyte;
}
```

The initial 16-bit values of the LFSR for the first 128 LFSR advances following a reset are listed below:

	0, 8	**1, 9**	**2, A**	**3, B**	**4, C**	**5, D**	**6, E**	**7, F**
00	FFFF	E817	0328	284B	4DE8	E755	404F	4140
08	4E79	761E	1466	6574	7DBD	B6E5	FDA6	B165
10	7D09	02E5	E572	673D	34CF	CB54	4743	4DEF
18	E055	40E0	EE40	54BE	B334	2C7B	7D0C	07E5
20	E5AF	BA3D	248A	8DC4	D995	85A1	BD5D	4425
28	2BA4	A2A3	B8D2	CBF8	EB43	5763	6E7F	773E
30	345F	5B54	5853	5F18	14B7	B474	6CD4	DC4C
38	5C7C	70FC	F6F0	E6E6	F376	603B	3260	64C2
40	CB84	9743	5CBF	B3FC	E47B	6E04	0C3E	3F2C
48	29D7	D1D1	C069	7BC0	CB73	6043	4A60	6FFA
50	F207	1102	01A9	A939	2351	566B	6646	4FF6
58	F927	3081	85B0	AC5D	478C	82EF	F3F2	E43B
60	2E04	027E	7E72	79AE	A501	1A7D	7F2A	2197
68	9019	0610	1096	9590	8FCD	D0E7	F650	46E6
70	E8D6	C228	3AB2	B70A	129F	9CE2	FC3C	2B5C
78	5AA3	AF6A	70C7	CDF0	E3D5	C0AB	B9C0	D9C1

An 8-bit value of 0 repeatedly encoded with the LFSR after reset produces the following consecutive 8-bit values:

	00	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**0A**	**0B**	**0C**	**0D**	**0E**	**0F**
00	FF	17	C0	14	B2	E7	02	82	72	6E	28	A6	BE	6D	BF	8D
10	BE	40	A7	E6	2C	D3	E2	B2	07	02	77	2A	CD	34	BE	E0
20	A7	5D	24	B1	9B	A1	BD	22	D4	45	1D	D3	D7	EA	76	EE
30	2C	DA	1A	FA	28	2D	36	3B	3A	0E	6F	67	CF	06	4C	26
40	D3	E9	3A	CD	27	76	30	FC	94	8B	03	DE	D3	06	52	F6
50	4F	88	80	95	C4	6A	66	F2	9F	0C	A1	35	E2	41	CF	27
60	74	40	7E	9E	A5	58	FE	84	09	60	08	A9	F1	0B	6F	62
70	17	43	5C	ED	48	39	3F	D4	5A	F5	0E	B3	C7	03	9D	9B
80	8B	0D	8E	5C	33	98	77	AE	2D	AC	0B	3E	DA	0B	42	7A
90	7C	D1	CF	A8	1C	12	EE	41	C2	3F	38	7A	0D	69	F4	01
A0	DA	31	72	C5	A0	D7	93	0E	DC	AF	A4	55	E7	F0	72	16
B0	68	D5	38	84	DD	00	CD	18	9E	CA	30	59	4C	75	1B	77
C0	31	C5	ED	CF	91	64	6E	3D	FE	E8	29	04	CF	6C	FC	C4
D0	0B	5E	DA	62	BA	5B	AB	DF	59	B7	7D	37	5E	E3	1A	C6
E0	88	14	F5	4F	8B	C8	56	CB	D3	10	42	63	04	8A	B4	F7
F0	84	01	A0	01	83	49	67	EE	3E	2A	8B	A4	76	AF	14	D5
100	4F	AC	60	B6	79	D6	62	B7	43	E7	E5	2A	40	2C	6E	7A
110	56	61	63	20	6A	97	4A	38	05	E5	DD	68	0D	78	4C	53
120	8B	D6	86	57	B2	AA	1A	80	18	DC	BA	FC	03	A3	4B	30

C Power Management

This appendix offers a system level overview of USB 3.0's power management features and capabilities. The following topics are also included:

- Examples of how end to end low power link state exit latencies are calculated
- A discussion of device-initiated link power management policies
- An example device implementation for Latency Tolerance Messaging
- System power considerations for SuperSpeed versus High Speed device interfaces

C.1 SuperSpeed Power Management Overview

The SuperSpeed architecture has been defined with platform power efficiency as a primary objective. Some of the key power efficiency enhancements include:

- Elimination of continuous device polling
- Elimination of broadcast packet transmission through hubs
- Introduction of link power management states enabling aggressive power savings when idle
- Host and device initiated transition to low power states
- Device and individual Function level suspend capabilities enabling devices to remove power from all, or only those portions of their circuitry that are not in use

C.1.1 Link Power Management

Link power management enables a link to be placed into a lower power state when the link partners are idle. The longer a pair of link partners remain idle, the deeper the power savings that can be achieved by progressing from U0 (link active) to U1 (link standby with fast exit), to U2 (link standby with slower exit), and finally to U3 (suspend).

After being configured by software, the U1 and U2 link states are entered and exited via hardware autonomous control. Hardware autonomous transitions for the U1 and U2 link states enable faster response times. This, in turn, translates to better power savings when entering a power saving state, and less impact on the operational state when exiting. The U3 link state however is entered only under software control, typically after a software inactivity timeout, and is exited either by software (host initiated exit) or hardware (remote wakeup). The U3 link state is directly coupled to the device's suspend state (refer to Section C.1.4).

C.1.1.1 Summary of Link States

Table C-1 provides a summary characterization of the SuperSpeed link states.

Table C-1. Link States and Characteristics Summary

Link State	Description	Characteristics	State Transition Initiator	Device Clock Gen On/Off	Typical Exit Latency Range
U0	Link active	Link operational state	N/A	On	N/A
U1	Link idle – fast exit	Rx and Tx circuitry quiesced	Hardware[1]	On or Off	µs
U2	Link idle – slower exit	Clock generation circuitry may additionally be quiesced	Hardware[1]	On or Off[2]	µs – ms
U3	Link suspend	Interface (e.g., Physical Layer) power may be removed	Entry: Software only Exit: Hardware or Software	Off	ms

Notes:

1. It is possible, under system test conditions, to instrument software initiated U1 and U2 state transitions.
2. From a power efficiency perspective it is desirable for devices to turn off their clock generation circuitry (e.g., their PLL) during the U2 link state.

C.1.1.2 U0 – Link Active

U0 is the fully operational, link active state. Packets of any type may be communicated over a link that is in the U0 State.

C.1.1.3 U1 – Link Idle with Fast Exit

U1 is a power saving state this is characterized by fast transition time back to the U0 State. Note that the predominant latency, when transitioning from the U1 → U0 state is imposed by the time that is required to achieve symbol lock between the two link partners.

C.1.1.3.1 U1 Entry

Either link partner for a given link can request a transition to the U1 link state. All downstream ports (hub or root ports) track inactivity using an inactivity timer mechanism. When a port's inactivity timer expires, if enabled, it requests transition to the U1 state. Upstream ports may also initiate U1 entry based on device specific policies.

When a port initiates U1 entry, its link partner may either accept or reject the request. The link level U0 → U1 transition process consists of one port transmitting an LGO_U1 link command, and its link partner responding with either an LAU (accept the request) or an LXU (reject the request) link command.

The most typical reason for rejecting a U1 transition request would be because the requesting port's link partner has some activity which will shortly require a packet transmission. Downstream ports would also reject a U1 transition request if not enabled to accept U1 transition requests. Rejection of a U1 transition request by an upstream port simply resets and restarts the requesting link partner's inactivity timer.

System software configures and then enables each device to initiate U1 entry. The primary programming parameters involved in setting up hardware autonomous link state management include:

U1DevExitLat – Parameter used by devices to report their maximum U1 to U0 exit latency (refer to Chapter 9 for details).

PORT_U1_TIMEOUT – Sets the value for the downstream port's U1 inactivity timer. When specifying a value between the range 0x01-0xFE it also enables the downstream port to send U1 entry transition requests to its link partner (refer to Chapter 10 for details).

U1_Enable – Enables an upstream port to initiate requests for transition into U1 (refer to Chapter 9 for details).

The U1_Enable feature controls whether that particular upstream port may initiate U1 entry. Regardless of whether the upstream port is enabled for U1 entry initiation or not, it still responds to requests for U1 entry from its link partner, either accepting or rejecting the transition request.

The following table illustrates the relationship between the downstream timeout and the upstream U1_Enable to configure the platform for any combination of port link state management. For example, if *U1_Enable* is enabled, and *PORT_U1_TIMEOUT* is set to FFH, then only the upstream port may initiate requests for transition to U1.

PORT_U1_TIMEOUT	U1_Enable	Initiating Port
01H-FEH	Disabled	Downstream Port only
FFH	Enabled	Upstream Port only
01H-FEH	Enabled	Either Port
0H or FFH	Disabled	Neither Port

For detailed information regarding the specifics of the U1 entry process, refer to Chapter 7 of this specification.

C.1.1.3.2 Exiting the U1 State

There are two ways to exit the U1 state. The link can return to the active U0 state or it can transition into a deeper power savings state (U2).

Transitioning from U1 → U0

Either link partner can initiate a transition from U1 → U0. This transition is normally initiated when a packet needs to be transmitted, such as an IN message from the host, or an ERDY message from a device. The transition process is initiated by first signaling a Low Frequency Periodic Signaling (LFPS) handshake. This is followed by link recovery and training sequences. Refer to Chapter 6 and 7 respectively.

Transitioning from U1 → U2

This transitions the link to an even lower power state and is triggered by a second inactivity timer (U2 inactivity timer). When a link enters U1, this starts the U2 inactivity timer. If the U2 inactivity timer expires while the link is still in U1, then both link partners transition silently from U1 → U2 without additional bus activity. The next sections discuss this in more detail.

C.1.1.4 U2 – Link Idle with Slow Exit

The purpose of the U2 link state is to use less power than the U1 state, however at the cost of increased exit latency. For example, clock generation circuitry may be quiesced in order to save additional power in comparison with U1. Under some implementation-specific circumstances, this may not make sense. For example, a hub may share one PLL across all of its ports so the PLL cannot be quiesced unless all of its ports are in U2.

The primary parameters involved in configuring a port for U2 link transitions include:

U2DevExitLat – Parameter used by devices to report their maximum U2 to U0 exit latency (refer to Chapter 9 for details).

PORT_U2_TIMEOUT – Sets the value of a downstream port's U2 inactivity timer. When specified with a value in the range 0x01-0xFE it also enables the downstream port to initiate U2 entry transition requests to its link partner (refer to Chapter 10 for details).

U2_Enable – Enables an upstream port to initiate requests for transitions into U2 (refer to Chapter 9 for details).

The U2_Enable feature controls whether an upstream port may initiate U2 entry from U0. Regardless of whether the upstream port is enabled for U2 entry initiation or not, it still responds to requests for U2 entry from its link partner, either accepting or rejecting the transition request.

The following table illustrates the relationship between the downstream timeout and the upstream U2_Enable to configure the platform for any combination of port link state management. For example, if *U2_Enable* is enabled, and *PORT_U2_TIMEOUT* is set to FFH, then only the upstream port may initiate requests for transition to U2.

PORT_U2_TIMEOUT	U2_Enable	Initiating Port
01H-FEH	Disabled	Downstream Port only
FFH	Enabled	Upstream Port only
01H-FEH	Enabled	Either Port
0H or FFH	Disabled	Neither Port

Transitioning from U1 → U2

U2 is typically entered directly from U1 as mentioned earlier. The U2 inactivity timer starts when a link enters U1, and when the U2 inactivity timer expires both link partners silently transition from U1 → U2.

Both link partners must be configured with the same U2 inactivity timeout value. This is done by first having software program the U2 inactivity timeout in the downstream port. The downstream port then sends an LMP containing the U2 inactivity timeout value to its link partner. Any changes to this value on the downstream port are also updated with the link partner in the same manner (refer to Chapter 10).

Note that U2 inactivity timer synchronization between link partners can never be perfect so there can be brief periods of time where one port is in U2 while its link partner is still in U1. However given that the U1 → U0 and U2 → U0 state transition processes are compatible with one another this corner condition is handled cleanly.

Transitioning directly to U2 from U0

A downstream port can be configured for direct, hardware autonomous transition from U0 → U2 by programming its U1 inactivity timer to zero while programming its U2 inactivity timer with a non-zero value in the range 0x01- 0xFE. In this case, when the U2 inactivity timer expires, the downstream port initiates U2 entry from U0.

When a port initiates U2 entry from U0, its link partner may either accept or reject the request. The link level U0 → U2 transition process consists of one port transmitting an LGO_U2 Link Command, and its link partner responding with either an LAU (accept the request) or an LXU (reject the request) Link Command.

Exiting U2

Exiting U2 can only result in a link state transition to U0. Either link partner can initiate U2 exit, which is initiated when a packet needs to be transmitted. The exit process is similar to that of a U1 → U0 transition, consisting of a Low Frequency Periodic Signaling handshake followed by link recovery and training.

C.1.1.5 U3 – Link Suspend

The U3 state is a deep power saving state where portions of device power may be removed, except as needed to perform the following functions:

- For upstream ports in devices and hubs:
  - Warm Reset signaling detection
  - Wakeup signaling detection (for host initiated wakeup)
  - Wakeup signaling transmission (for remote wakeup capable devices)
- For downstream ports in hubs and hosts:
  - Warm Reset generation
  - Disconnect event detection
  - Wakeup signaling detection (for remote wakeup)
  - Wakeup signaling transmission (for host initiated wakeup)

The purpose of U3 is to minimize power consumption during device or system suspend.

VBUS remains active during U3. Any power rail switching by devices is implementation specific and beyond the scope of this specification.

Entering the U3 State

U3 entry may only be initiated by the host (refer to Chapter 10 for details). Software typically implements an inactivity timeout for the purpose of placing a function into suspend following a long period of inactivity.

When a downstream port initiates a U3 entry request, its link partner is not allowed to reject it. The downstream port sends an LGO_U3 Link Command, and its link partner responds with an LAU Link Command (refer to Chapter 7 for details).

Exiting the U3 State

The only legitimate link state transition from U3 is U3 → U0, and either link partner can initiate it.

Host software initiates U3 exit on a downstream port by issuing a SetPortFeature(PORT_LINK_STATE) request to the desired downstream port. Upstream ports (e.g., upstream port of a hub, or a peripheral device), initiate U3 exit in response to a remote wakeup event. An example of this would be an incoming Wake on LAN packet for a USB attached network interface device. The exit process consists of a Low Frequency Periodic Signaling (LFPS) handshake followed by link recovery and training.

Device Initiated U3 Exit

If the exit was initiated by a device, the specific function within the device that initiated the wakeup would follow up the LFPS triggered transition to U0 by sending a Function Wake device notification packet to the host.

Host initiated U3 Exit

For host initiated U3 exit, the LFPS handshake process allows devices up to 20 ms to complete, allowing sufficient time for a device to turn on a switched power rail if implemented (refer to Chapter 6 for details).

The software interface associated with U3 consists of a set of port controls and function controls. The port controls, e.g., initiating U3 on a hub downstream port, are described in Section C.1.2.3. The function controls, e.g., enabling a function (device) for remote wakeup, are described in Section C.1.4.1.

C.1.2 Link Power Management for Downstream Ports

Hubs play several critical roles in link power management. They perform the following functions:

- Coordinate the upstream port link power management state with that of their downstream ports.
- Handle packet deferral, where the hub tells the host that a packet was sent to a downstream port that is not currently in U0.
- Provide inactivity timers on downstream ports to initiate U1 and U2 entry.

C.1.2.1 Link State Coordination and Management

A hub monitors its downstream ports' link states and keeps its upstream port in the lowest power link state it can without allowing it to be in a lower power state than any of its downstream ports. The intent for this policy is to ensure that the path to the host, (i.e., the upstream port), is as active as the most active of its downstream ports.

When a device initiates a transition from a low power state back to U0 on a hub downstream port, the hub begins transitioning its upstream port to U0 immediately, in parallel with the downstream port.

For packets traveling downstream, hubs must first receive the packet, determine which of its downstream ports the packet is targeted at, and then only initiate a transition to U0 for that particular downstream port.

USB 3.0 hubs use a unicast packet transmission model and this improves the power efficiency of the platform in every instance where a hub is deployed.

C.1.2.2 Packet Deferring

Packet deferring is a mechanism that enables efficient bus utilization while supporting aggressive link power management. Packet deferring achieves this by enabling a hub to respond on behalf of a downstream device whose link is in a low power state. This allows the host to make forward progress while the device is brought back to the active state.

Hub's Role in Packet Deferring

After receiving a header packet from the host and detecting a packet deferring condition, the hub informs the host of this by sending a deferred header packet back to the host. The hub also sends the original header packet to the device, with the deferred field asserted, once it is brought back to the U0 state. The host treats this deferred header packet as it would receipt of an NRDY, and so is then free to initiate transfers with other devices' endpoints instead of waiting for the sleeping link to return to U0 (refer to Chapter 10 for details).

Device's Role in Packet Deferring

When a device receives an IN or an OUT header packet with the deferred field asserted, it prepares for the transfer, sends an ERDY to the host when ready, and keeps its link in U0 until the transfer occurs.

The host ultimately responds to the ERDY by rescheduling the original transfer.

C.1.2.3 Software Interface

The software interface for downstream port power management consists of the following port controls and status fields:

- PORT_LINK_STATE feature and port status field
- PORT_REMOTE_WAKE_MASK feature
- C_PORT_LINK_STATE port status change bit
- PORT_U1_TIMEOUT feature
- PORT_U2_TIMEOUT feature

The PORT_LINK_STATE

This feature is used to request a link state change from any current U-state to any next U-state. In a normal operating environment, this feature is used solely to request U0 → U3 and U3 → U0 transitions. It can be used for test purposes though, to request other state transitions.

The PORT_REMOTE_WAKE_MASK

This feature is used to mask each remote wakeup event that might be originated at a downstream port.

Note that if remote wake notifications for connect, disconnect, or over current events are disabled, these events are still captured and reported as port status change events after the host or hub is resumed.

C_PORT_LINK_STATE

This flag is used to signal completion of a transition from U3 → U0. Specifically, assertion of this flag results from a host initiated wakeup on a downstream port.

Once the C_PORT_LINK_STATE flag is set, a port status change event is sent to system software indicating that the downstream port and its link partner have completed the transition to the U0 state.

Note that C_PORT_LINK_STATE is not asserted in the event of a remote wakeup. As discussed previously, in the event of a Remote Wakeup the associated function sends the host a Function Wake device notification packet.

PORT_U1_TIMEOUT

This feature is used to enable and disable U1 entry on downstream ports. It also specifies the U1 inactivity timeout value.

Timeout Value	U1 Transition Capabilities	U1 Transition Initiation
00H	U1 transitions disabled; will not initiate or accept link partner requests	n/a
01H-FEH	U1 transitions enabled; port will initiate and accept link partner requests	Following timer expiration using specified Timeout Value
FFH	U1 transition initiation disabled; will accept transition requests from its link partner	n/a

PORT_U2_TIMEOUT

This feature is used to enable and disable U2 on downstream ports, and also to set an inactivity timeout for initiating a transition to U2.

Timeout Value	U2 Transition Capabilities	U2 Transition Initiation
00H	U2 transitions disabled; will not initiate or accept link partner requests	n/a
01H-FEH	U2 transitions enabled; port will initiate and accept link partner requests	Following timer expiration using specified Timeout Value
FFH	Direct U1 → U2 transition is disabled; will accept U2 transition requests (if current link state is U0) from its link partner	n/a

Other hub port controls can impact link power management behavior, e.g., the PORT_RESET feature, but are not covered here (refer to Chapter 10 for details).

C.1.3 Other Link Power Management Support Mechanisms

C.1.3.1 Packets Pending Flag

Devices may use the Packets Pending flag (refer to Chapter 8) to help decide when to place their link in a low power state. The Packets Pending flag provides an indication of whether the host controller has any additional packets to transfer on the schedule associated with a given non-Stream endpoint. When there are no more packets pending for all non-Stream endpoints on a device, the device may place its link in a low power state immediately.

For Stream endpoints, the Packet Pending flag is an indication of whether the host controller has any additional packets to transfer on the schedule associated with a given Stream. When there are

no more packets pending for any Streams and for all endpoints on a device, the device may place its link in a low power state immediately.

C.1.3.2 Support for Isochronous Transfers

If a link is in a low power state when an isochronous transfer is scheduled, the latency to transition to U0 from source to destination could potentially delay the transfer beyond its subscribed isochronous service interval.

To ensure that isochronous service contract guarantees are satisfied, a SuperSpeed mechanism (Ping) has been defined to bring all paths between the host and an isochronous endpoint to U0 as a routine step in servicing isochronous endpoints.

The host controller, with sufficient information to know how long any given path might take to become fully active, factors this link path exit latency into its isochronous service scheduler and uses the Ping protocol to ensure that the links are brought to a fully active state in time to meet the isochronous service contract.

The ping process consists of the following:

- The host sends a PING packet to a device.
- Hubs route the PING packet toward the targeted device.
- The device responds to the PING packet by sending a PING_RESPONSE packet to the host.
- The device keeps its link in U0 until it receives a subsequent packet from the host.

After sending a PING packet to one device and prior to receiving a PING_RESPONSE packet, the host may transfer data with other devices. Much like the Packet Deferring mechanism, the Ping mechanism enables efficient bus utilization while at the same time supporting significant power savings.

C.1.3.3 Support for Interrupt Transfers

If any links between the host and a scheduled interrupt endpoint are in a low power state, the latency to transition the end to end pathway to U0 could potentially delay the transfer beyond the subscribed interrupt service interval. A host controller, possessing knowledge of link path exit latency between itself and any given device within the link hierarchy, is able to schedule interrupt transfers far enough in advance to compensate for these latencies.

C.1.4 Device Power Management

Device power management is directed primarily under software control, with various hardware mechanisms to support it. Device power management consists of some function level mechanisms plus some device and hub mechanisms.

C.1.4.1 Function Suspend

A function may be placed into function suspend independent of other functions using the FUNCTION_SUSPEND feature. The FUNCTION_SUSPEND feature is also used to enable function remote wakeup (refer to Chapter 9 for details).

If a composite device has at least one of its functions in suspend while other functions remain active, i.e., the device's upstream port is not in U3, a mechanism is needed for a suspended function

to signal a remote wakeup. This is done with the Function Wake device notification packet (refer to Chapter 8 for details).

C.1.4.2 Device Suspend

Device suspend is a device-wide state entered and exited intrinsically as a result of a device's upstream port entering and exiting the U3 state. A device may be transitioned into device suspend regardless of the function suspend state of any function within the device.

Devices may implement a switched power rail and remove power from large portions of the device while in suspend. Some device state information must be retained in a persistent state during suspend (refer to Chapter 9 for details).

C.1.4.3 Host Initiated Suspend

Suspending a Device

The host transitions a device into the suspend state according to the following sequence:

- Enable remote wakeup, if needed
- A SetPortFeature(PORT_LINK_STATE, U3) request is issued to the downstream port of which the targeted device is its link partner.
- The downstream port initiates U3 entry by sending an LGO_U3 link command.
- The device sends an LAU link command (acceptance is non-negotiable).
- The downstream port sends an LPMA link command.
- Both link partners transition their transmitters to electrical idle and enter the U3 state (refer to Chapter 7 for details).

Suspending the USB Link Hierarchy

A link hierarchy of devices and hubs is placed into suspend on a device by device basis under host software control. First, all peripheral devices in the hierarchy are placed into suspend. Then the hubs connected to the peripheral devices are placed into suspend. This is repeated all the way up the hierarchy until reaching the root ports of the USB hierarchy.

Note that by using the same technique a select subset of a given USB link hierarchy could be suspended rather then the whole of it if so desired.

C.1.4.4 Host Initiated Wake from Suspend

Host initiated wake from suspend (U3 → U0) of an individual device, group of devices, or of the entire link hierarchy is accomplished using the same repetitive process, one link at a time.

Figure C-1 illustrates the Host initiated Wake Sequence.

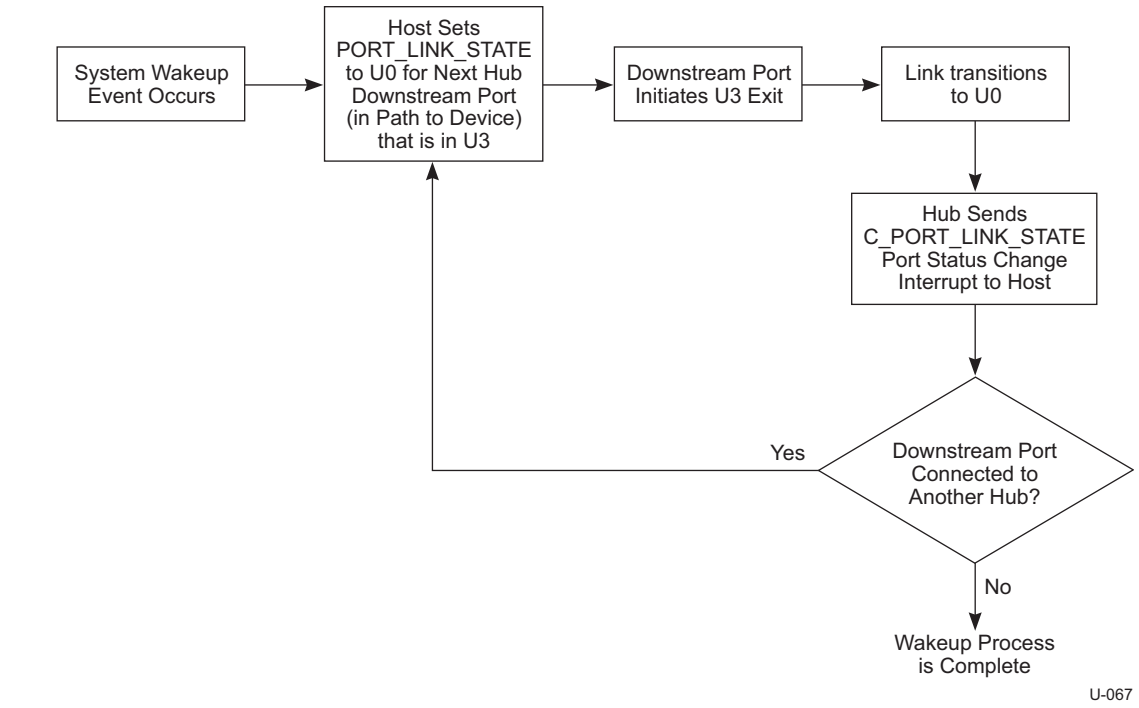


Figure C-1. Flow Diagram for Host Initiated Wakeup

C.1.4.5 Device Initiated Wake from Suspend

A device initiated transition from suspend (U3 → U0) follows the sequence outlined below:

1. The device transmits LFPS wakeup signaling to its link partner.
2. The LFPS signaling is propagated upstream until it reaches the root hub or a hub that is not in U3. This hub is referred to as the Controlling Hub.
3. The Controlling Hub then automatically reflects LFPS wakeup signaling on the downstream port which had received (from the opposite direction) the wakeup signaling.
4. Each hub in the direct path to the remote wakeup device propagates the wakeup signaling downstream on the hub downstream port that had received wakeup signaling. As the wakeup signaling is propagated downstream, each link completes the LFPS handshake and transitions to U0 (refer to Chapters 6 and 7 for details).
5. After all of the links between the Controlling Hub and the remote wakeup device transition to U0, the function within the remote wakeup device that had originated the remote wakeup sends a Function Wake device notification packet to the host. This in turn causes a software interrupt, and in the service of this interrupt the function suspend state is cleared for that function.

C.1.5 Platform Power Management Support

The Latency Tolerance Message (LTM) feature allows a platform to make dynamic tradeoffs between power and performance. It enables this, in cooperation with devices, without imposing additional cost.

The LTM protocol enables USB devices to inform the host how long they can tolerate lack of service before experiencing unintended side effects. Each device provides this information in the form of a Best Effort Latency Tolerance (BELT) value. A given device's BELT value is derived considering all configured endpoints, typically conforming to the endpoint with the lowest latency tolerance.

LTM provides the ability for a device to dynamically change its BELT value to more accurately reflect, for example, long periods of anticipated idle time. The platform can potentially take advantage of this insight and, along with other system-related information, conserve more energy at the system level without running the risk of unintended side effects.

C.1.5.1 System Exit Latency and BELT

A device's reported BELT value has to comprehend not only its own intrinsic design characteristics, such as its internal buffering, but also factor in other associated end to end latencies between itself and the host. These would include other latencies associated with the time required to awaken sleeping links, the number of hubs between the device and the host, host processing time, packet propagation delays, etc. The system provides the device with additional system latency information, through the SET_SEL request, such that the device's intrinsic BELT value can be adjusted downward to account for these other factors. Refer to Chapter 8 for detailed LTM specification and to Chapter 9 for specification details regarding the SET_SEL request.

Device implementation determines the total latency that a device can tolerate. The primary factors are the amount of data that the device is required to produce or consume, and the amount of buffering on the device. The total device latency tolerance must be allocated among different system components.

Figure C-2 illustrates the total latency a device may experience within the context of LTM. The latency is the sum of parameters t1, t2, t3, and t4:

- t1: the time to transition all links in the path to the host to U0 when the transition is initiated by the device
- t2: the time for the ERDY to traverse the interconnect hierarchy from the device to the host
- t3: the time for the host to consume the ERDY and transmit a response to that request
- t4: the time for the response to traverse the interconnect hierarchy from the host to the device

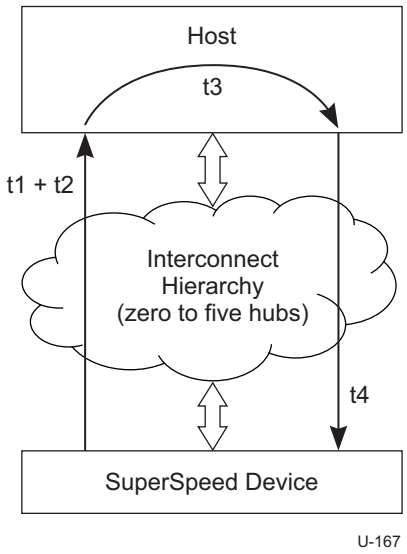


Figure C-2. Device Total Intrinsic Latency Tolerance

Devices may calculate their BELT value by subtracting U1SEL or U2SEL (refer to the SET_SEL request in Chapter 9) from their total intrinsic latency tolerance. If the device allows its link to enter U2, then the device calculates its BELT by subtracting U2SEL from its total intrinsic latency tolerance. If the device does not allow its link to enter U2 but does allow its link to enter U1, then the device calculates its BELT by subtracting U1SEL from its total intrinsic latency tolerance.

U1SEL and U2SEL are calculated and programmed by host software. Example calculations for t1 are provided in Section C.2. For LTM purposes t2 and t4 should be calculated by host software as follows:

- For t2, a hub may delay forwarding the ERDY by up to one maximum packet size (approximately 2.1 µs including framing) when there is a transfer in progress. Each additional hub will delay forwarding the ERDY by up to approximately tHubDelay to transfer the packet. The value of t2 is determined as follows:
  - If there are zero hubs in the direct path between the device and the host, then t2 is zero
  - If there are one or more hubs in the direct path between the device and the host, then t2 is approximately 2.1 µs + tHubDelay * (number of hubs – 1)
- For t4, a hub may delay forwarding a packet by up to approximately tHubDelay. The value of t4 is approximately tHubDelay times the number of hubs in the direct path between the device and the host).

C.1.5.2 Maximum Exit Latency and PING

The host controller is provided with a Maximum Exit Latency (MEL) value that it uses to schedule a PING relative to a periodic transfer. The Maximum Exit Latency must comprehend worst case round trip delay of sending a PING to a device and receiving the PING_RESPONSE from it.

The Maximum Exit Latency factors in the end to end latencies between host and the device. These would include other latencies associated with the time required to awaken sleeping links, the

number of hubs between the host and the device, device processing time, packet propagation delays, etc.

The Maximum Exit Latency (tMEL) is the sum of parameters tMEL1, tMEL2, tMEL3, and tMEL4.

C.1.5.2.1 Maximum Exit Latency t1 (tMEL1)

The tMEL1 delay is the time to transition all links in the path to the device to U0 when the transition is initiated by the host. The method for calculating MEL t1 delay is described in Sections C.2.1.1 and C.2.2.1.

C.1.5.2.2 Maximum Exit Latency t2 (tMEL2)

The tMEL2 delay is the time for a PING TP to traverse the interconnect hierarchy from the host to the device. tMEL2 is the sum of the tHubDelay values for each hub in the path, and the TP Propagation Delay (i.e., 20 symbol times or 40 ns at 5 Gb/s) across each link in the path, and for a 5 Gb/s signaling rate is calculated as:

tMEL2 = (sum of wHubDelay values) + (40 ns * (number of hubs + 1)).

Where, a wHubDelay value is provided by the SuperSpeed Hub Descriptor of each hub in the path, respectively.

C.1.5.2.3 Maximum Exit Latency t3 (tMEL3)

The tMEL3 delay is the time for the device to receive the PING and generate the PING_RESPONSE, which is defined by tPingResponse. Refer to Table 8-33.

C.1.5.2.4 Maximum Exit Latency t4 (tMEL4)

The tMEL4 delay is the time for the PING_RESPONSE to traverse the interconnect hierarchy from the device to the host. Since wHubDelay defines the downstream and upstream delay through a hub, the propagation delay of a PING_RESPONSE upstream is identical to that of a PING downstream delay (tMEL2), with one exception. In the upstream path a TP may be queued behind a Max Packet Size DP, so an additional 2.1 µs of delay is included to comprehend the "congestion jitter". tMEL4 is calculated as:

tMEL4 = tMEL2 + 2.1 µs.

C.2 Calculating U1 and U2 End to End Exit Latencies

This section provides examples of how to calculate the exit latency (i.e., time to transition from a non-U0 state to a U0 state) spanning the end to end path between a device and the host. Examples are given for both device initiated exit and host initiated exit.

Figure C-3 depicts a SuperSpeed hierarchy and calls out the relevant parameters used in the calculations that follow.

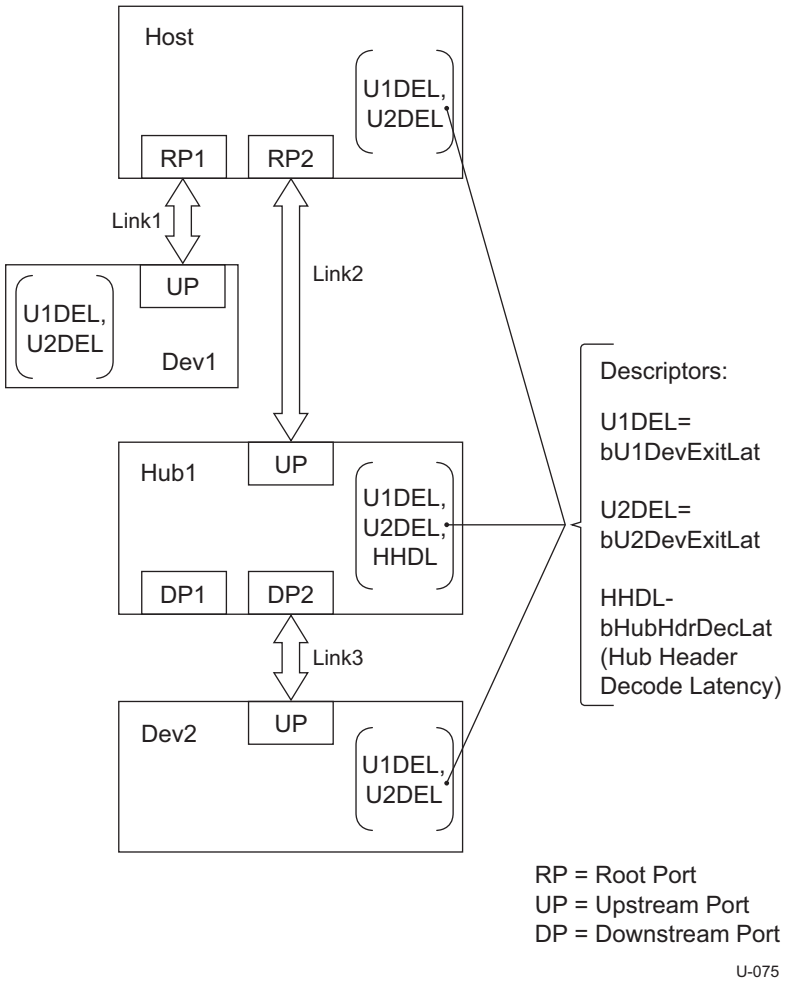


Figure C-3. Host to Device Path Exit Latency Calculation Examples

C.2.1 Device Connected Directly to Host

C.2.1.1 Host Initiated Transition

In this example, a peripheral device (Dev1) and a host controller root port (RP1) are link partners communicating over a link (Link1).

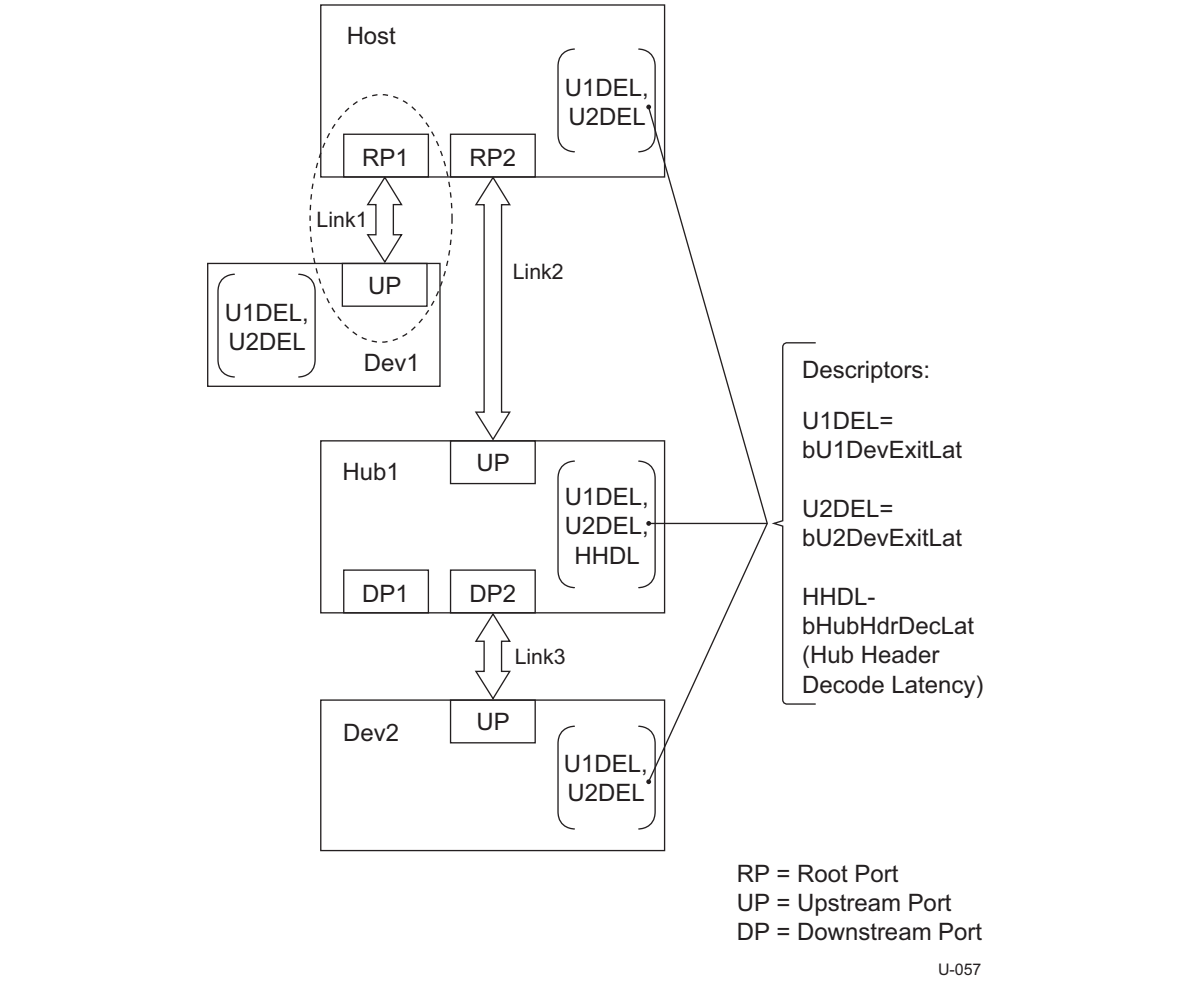


Figure C-4. Device Connected Directly to a Host

U1 → U0 Transition Latency

The host initiates the transition by transmitting LFPS. At this point, both link partners' transitions from U1 → to U0 are executed in parallel.

The exit latency is characterized by the largest of the two device exit latencies: Dev1:U1DEL and RP1:U1DEL

U2 → U0 Transition Latency

In this example it is assumed that at least one of the link partners (e.g., the RP1) is enabled for U2. The host initiates the transition by transmitting LFPS. At this point both link partners execute transition from U2 → to U0 in parallel.

The exit latency is characterized by the largest of the two device exit latencies: Dev1:U2DEL and RP1:U2DEL

C.2.1.2 Device Initiated Transition

These transition latencies are the same as for the host initiated cases.

- U1 End to End Exit Latency is **the larger of Dev1:U1DEL and RP1:U1DEL**
- U2 End to End Exit Latency is **the larger of Dev1:U2DEL and RP1:U2DEL**

C.2.2 Device Connected Through a Hub

In this example a peripheral device (Dev2) is connected to one of a hub's downstream ports (DP2) via a link (Link3), which in turn is connected to a host controller's root port (RP2) via a link (Link2).

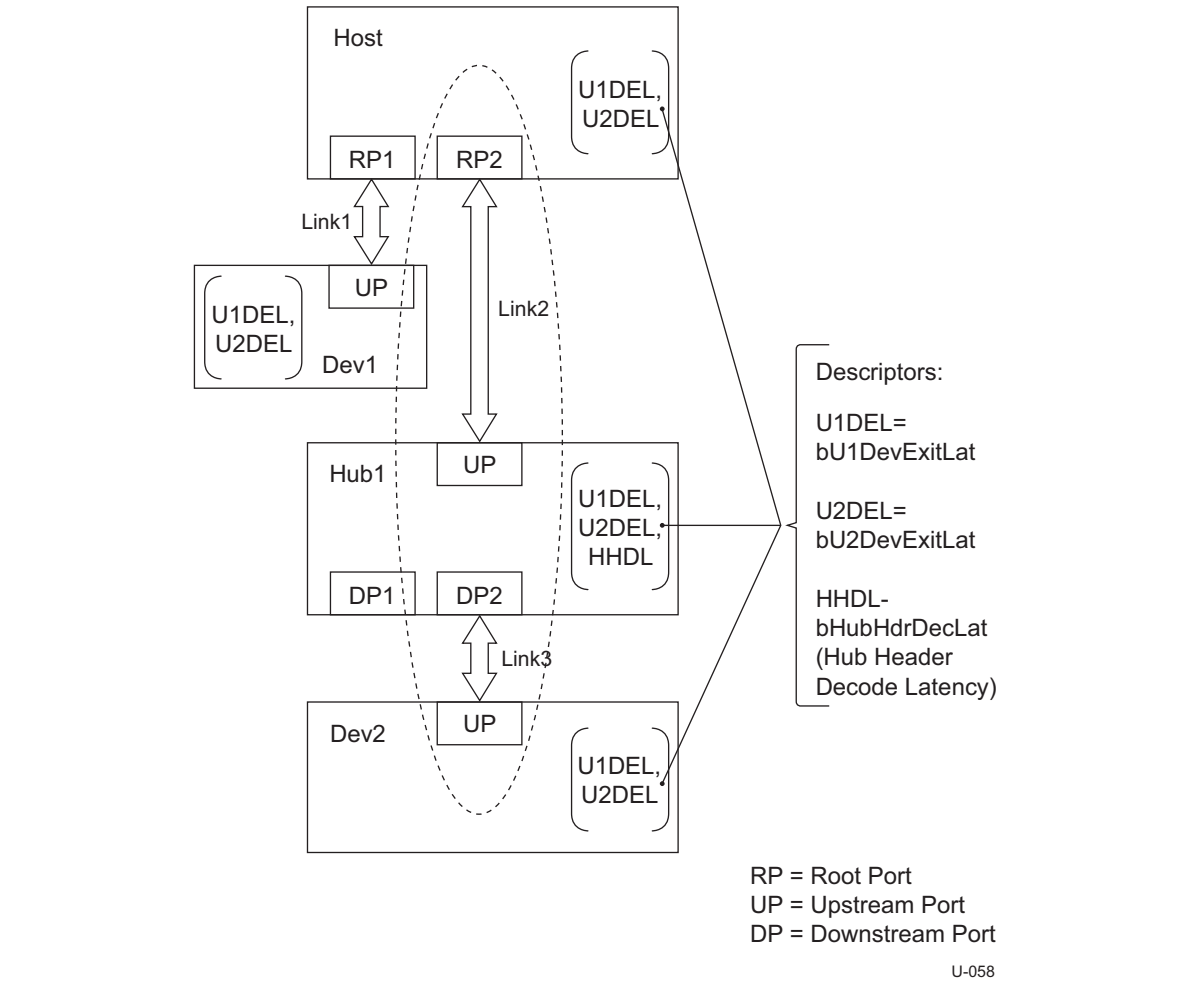


Figure C-5. Device Connected Through a Hub

C.2.2.1 Host Initiated Transition

U1 → U0 Transition Latency

This example highlights the end to end latency incurred when transitioning both Link2 and Link3 from U1 → U0. For the purposes of this example, it is assumed that all link partners are enabled for U1, and that both Link2 and Link3 are currently in the U1 state.

Figure C-6 shows the chronological sequence of events. Following the figure is a more detailed description of each stage of the multi-hop link state transition.

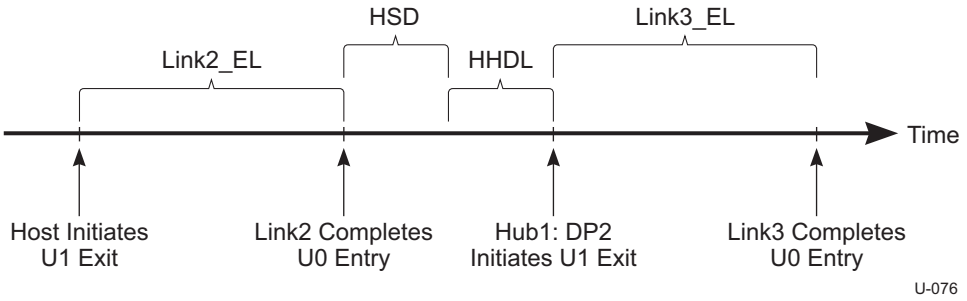


Figure C-6. Downstream Host to Device Path Exit Latency with Hub

1. The host is prepared to send a packet to Dev2, however it first needs to bring Link2 out of U1 before it is able to send the packet. The host begins the process by transmitting LFPS on RP2 to Hub1's upstream port (UP) which then starts both link partners transitioning in parallel towards U0. This latency is characterized by the larger of ***RP2:U1DEL*** and ***Hub1:U1DEL***
2. Once the Link2 partners are in U0, the host schedules the packet targeting Dev2. After a Host Scheduling Delay (HSD) the packet is then sent over Link2 where it then is routed to Link3. This routing incurs the latency associated with the Hub having to parse the packet header to determine the target downstream port for the packet. This latency is characterized by the hub parameter HHDL.
3. The final hop requires Hub1:DP2 to signal LFPS over Link3 at which point the final component of the end to end latency is executed by the Link3 partners in parallel. This final ingredient to the end to end latency is characterized by the larger of ***Hub1:U1DEL*** and ***Dev2_UP:U1DEL***.

The total latency for end to end link transition to U0 can be summarized as:

Max(RP2:U1DEL, Hub1:U1DEL) + HSD + HUB1:HHDL + Max(Hub1:U1DEL, Dev2_UP:U1DEL)

U2 → U0 Transition Latency

This example highlights the end to end latency incurred when transitioning both Link2 and Link3 from U2 → U0. For the purposes of this example it is assumed that all link partners are enabled for U2 and that both Link2 and Link3 are currently in the U2 state.

1. The host is prepared to send a packet to Dev2, however it first needs to bring Link2 out of U2 before it is able to send the packet. The host begins the process by transmitting LFPS to Hub1:UP which then starts both link partners transitioning in parallel towards U0. This latency is characterized by the larger of ***RP2:U2DEL*** and ***Hub1:U2DEL***.
2. Once the Link2 partners are in U0 the host sends its packet targeting Dev2 over Link2 where it then needs to be routed to Link3. This incurs the latency associated with the Hub having to parse the packet header to determine the target downstream port for the packet. This latency is characterized by the hub parameter ***HHDL***.

3. The final hop requires Hub1:DP2 to signal LFPS over Link3 at which point the final component of the end to end latency is executed by the Link3 partners in parallel. This final ingredient to the end to end latency is characterized by the larger of ***Hub1:U2DEL*** and ***Dev2_UP:U2DEL***.

The total exit latency for end to end link transition to U0 can be summarized as:

Max(RP2:U2DEL, Hub1:U2DEL) + HSD + Hub1:HHDL + Max(Hub1:U2DEL, Dev2_UP:U2DEL)

C.2.2.2 Device Initiated Transition

This section provides some examples for calculating end to end exit latencies for device initiated exit.

U1 → U0 Transition Latency

This example highlights the end to end latency incurred when transitioning both Link2 and Link3 from U1 → U0. For the purposes of this example it is assumed that all link partners are enabled for U1 and that both Link2 and Link3 are currently in the U1 state.

Figure C-7 depicts the end to end link state transition in chronological order. Following the figure a more detailed description of the sequence is provided.

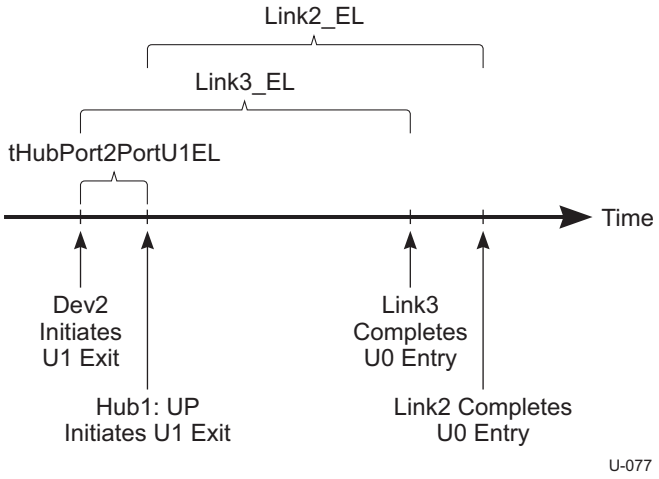


Figure C-7. Upstream Device to Host Path Exit Latency with Hub

1. Dev2 is prepared to send a packet upstream to the host. However, it first needs to bring Link3 out of U1 and back to U0 before it is able to send the packet. Dev2 begins the process by transmitting LFPS to Hub1:DP2 which then starts both link partners transitioning in parallel towards U0. The Link3 exit latency (**Link3_EL**) is characterized by the larger of ***Dev2_UP:U1DEL*** and ***Hub1_DP2:U1DEL***.
2. After a latency of ***tHubPort2PortU1EL***, the time it takes the hub to determine that one of its downstream ports is awakening, the hub then begins signaling LFPS on Link2 to initiate transition of the Link2 partners (hub's upstream port and host controller root port RP2) to U0.

3. The last component of the end to end exit latency is characterized by the larger of ***Hub1_UP:U1DEL*** and ***RP2:U1DEL*** which represents the Link2 exit latency (**Link2_EL**).

End to end exit latency = Max(Link3_EL, (Link2_EL + tHubPort2PortU1ExitLat))

U2 → U0 Transition Latency

This example highlights the end to end latency incurred when transitioning both Link2 and Link3 from U2 → U0. For the purposes of this example it is assumed that all link partners are enabled for U2 and that both Link2 and Link3 are currently in the U2 state.

1. Dev2 is prepared to send a packet upstream to the host. However, it first needs to bring Link3 out of U2 and back to U0 before it is able to send the packet. Dev2 begins the process by transmitting LFPS to Hub1:DP2 which then starts both link partners transitioning in parallel towards U0. The Link3 exit latency (**Link3_EL**) is characterized by the larger of ***Dev2_UP:U2DEL*** and ***Hu1b_DP2:U2DEL***.
2. After a latency of ***tHubPort2PortU2EL***, the time it takes the hub to determine that one of its downstream ports is awakening, the hub then begins signaling LFPS on Link2 to initiate transition of the Link2 partners (hub's upstream port and host controller root port RP2) to U0.
3. The last component of the end to end exit latency is characterized by the larger of ***Hub1_UP:U2DEL*** and ***RP2:U2DEL*** which represents the Link2 exit latency (**Link2_EL**).

End to end exit latency = Max(Link3_EL, Link2_EL + tHubPort2PortU2EL)

C.3 Device-Initiated Link Power Management Policies

Power savings resulting from the effective use of link power management can have a significant impact on system power consumption. For example, without using link power management, the average battery life of a typical notebook computer could be decreased by as much as 15%.

Both devices and downstream ports can initiate U1 and U2 entry.

- **Downstream ports** have inactivity timers used to initiate U1 and U2 entry. Downstream port inactivity timeouts are programmed by system software.
- **Devices** may have additional information available that they can use to decide to initiate U1 or U2 entry more aggressively than inactivity timers.

This section describes policies for devices to initiate U1 or U2.

C.3.1 Overview and Background Information

Devices can save significant power by initiating U1 or U2 more aggressively rather than waiting for downstream port inactivity timeouts. For example, an isochronous device may substantially increase U1 residency by initiating U1 upon completion of isochronous transfers within each service interval.

Devices can use the following information to help determine when to initiate U1 or U2 due to endpoint idle conditions:

- The type of device endpoints and related flags (refer to Chapter 8)
  - — Packets Pending flag, used with bulk endpoints
  - — End of Burst flag, used with interrupt endpoints
  - — Last Packet flag, used with isochronous endpoints

- Protocol level endpoint flow control conditions, e.g., an endpoint having sent an NRDY
- Device class and device implementation
- U1 and U2 device-to-host exit latencies

U1 and U2 device-to-host exit latencies are the total latency to transition all links in the path between the device and the host to U0, when exit is initiated by the device. The device may assume a device-to-host exit latency based on the worst case exit latency (device connected five hubs deep). The device may alternatively use the device-to-host exit latency provided with the U1PEL and U2PEL fields of the SET_SEL request (refer to Chapter 9).

C.3.2 Entry Conditions for U1 and U2

A device should initiate U1 or U2 when idle conditions are met for all its endpoints. A device typically initiates U1. However, if a device is able to determine that its link will not be needed for a long time, then the device may be able to initiate U2. For example, WiFi has a protocol where its radio may be shut off for long periods, e.g., 100 ms, and since the link is not needed during this time it may be placed in U2.

A device should initiate U2 if it is able to determine its link is not needed for a period of time that exceeds the U2 device-to-host exit latency (plus an appropriate guard band). The device should initiate U1 in all other cases.

Devices should consider the device-to-host exit latency when determining whether to initiate U1 and U2 entry. The host-to-device exit latency and the device-to-host exit latency are both considered by host software when determining whether to enable U1 or U2 on each link. Devices are enabled to initiate U1 and U2 with the U1_Enable and U2_Enable feature selectors (refer to Chapter 9).

The following subsections offer recommendations for determining when an endpoint is idle, or does not need to use the link for a known period, based on endpoint type. Idle conditions may be determined in other implementation specific ways.

C.3.2.1 Control Endpoints

A control endpoint is idle when all of the following conditions are met:

- Device is in the configured state
- Device is not in the midst of a control transfer
- Either an NRDY was sent, or the Packets Pending flag was set to zero in the last ACK packet received from the host
- Device does not have a pending ERDY

C.3.2.2 Bulk Endpoints

A bulk endpoint is idle when both of the following conditions are met:

- Either an NRDY was sent, or the Packets Pending flag was set to zero in the last ACK packet received from the host
- Device does not have a pending ERDY

Some devices can also determine that their link is not needed for a known period of time. For example, a mass storage device may need to spin up a spindle to service a request. Since the spin up time can be hundreds of milliseconds, the device should place its link in U2.

After a device has sent an ERDY associated with a bulk endpoint, the link should be kept in U0 until the host sends a request in response to the ERDY (or until the tERDYTimeout occurs, refer to Section 8.13).

C.3.2.3 Interrupt Endpoints

An interrupt endpoint is idle when both of the following conditions are met:

- Either an NRDY was sent, or the Packets Pending flag was set to zero in the last ACK packet received from the host.
- Device does not have a pending ERDY.

After a device has sent an ERDY associated with an interrupt endpoint, the link should be kept in U0 until the host sends a request in response to the ERDY (or until the tERDYTimeout occurs) in order to achieve the subscribed interrupt service latency. However, when all transfers for a given service interval have been completed, the endpoint will not need the link until the next service interval. The device may be able to place its link in U1 or U2 during this time. The End of Burst flag can be used to determine when all transfers for a given service interval are completed. Note that hosts are required to initiate interrupt transfers far enough ahead of a transfer window to meet subscribed service requirements.

C.3.2.4 Isochronous Endpoints

An isochronous endpoint is idle when all transfers for a given service interval have been completed, as indicated by the Last Packet flag. The endpoint will not need the link until the next service interval. Note that the host is required to send a PING packet far enough ahead of a transfer window to meet subscribed service requirements.

C.3.2.5 Devices That Need Timestamp Packets

When a device needs timestamp information, it needs to ensure that its link is in U0 when the next bus interval boundary is reached in order to receive a timestamp packet. If the device's link is not in U0, it should transition to U0 prior to the next bus interval boundary. The device must track when the bus interval boundary will occur. The device initiates a transition to U0 a period of time before the bus interval boundary occurs, where the period of time is the device-to-host link exit latency.

C.4 Latency Tolerance Message (LTM) Implementation Example

Computer systems typically maintain a high state of readiness to service devices even when the computer system is idle. LTM supports a mechanism for a system to reduce its state of readiness with the cooperation of USB 3.0 devices. This may result in substantial system power savings without requiring additional cost to devices.

This section provides a device implementation example for LTM support. This example is based on a model using two device Latency Tolerance states, an active state and an idle state. Each state has a different Best Effort Latency Tolerance (BELT).

In the following subsections, first a description of BELT and its relationship to overall system exit latency is given. This is followed by a description of a device state machine implementation example.

C.4.1 Device State Machine Implementation Example

This section describes an example of a typical device implementation. It assumes the device implementation supports both U1 and U2 in conjunction with LTM.

In this example, two device Latency Tolerance states (LT-states) are defined:

- **LT-idle state**: the device is idle and can tolerate a larger latency from the system (this is the default state).
- **LT-active state**: the device has determined a need to perform data transfers with the host and wants a shorter latency from the system.

A state machine is illustrated in Figure C-8.

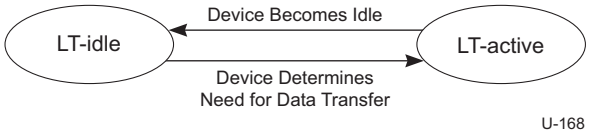


Figure C-8. LT State Diagram

The device described by this implementation example is designed to accommodate the worst case value for U1SEL during LT-active, and the worst case value for U2SEL during LT-idle.

The following device design goals are to be met:

- Design for a minimum LTM BELT of 1 ms when in LT-idle
- Design for a minimum LTM BELT of 125 µs when in LT-active

C.4.1.1 LTM-Idle State BELT

The device determines its LT-idle state BELT value by subtracting U2SEL from the total latency it can tolerate. To achieve a minimum LT-idle state BELT of 1 ms, the total latency the device must be able to tolerate is 1 ms plus the worst case value for U2SEL, or a total of approximately 3.1 ms (refer to Section C.1.5.1). The worst case U2SEL is based on a worst case device-to-host U2 exit latency of 2.053 ms for t1 (2.047 ms device exit latency plus 1 µs for each of five hubs), plus 0.003 ms for t2, plus 0.001 ms for t4, plus some guard band.

For system implementations where U2SEL is less than its worst case value, the device reports a BELT value larger than 1 ms.

C.4.1.2 LTM-Active State BELT

The device determines its LT-active state BELT value by subtracting U1SEL from the total latency it can tolerate. To achieve a minimum LT-active state BELT of 125 µs, the total latency the device must be able to tolerate is 125 µs plus the maximum value for U1SEL, or a total of approximately 145 µs. The worst case U1SEL is based on a worst case device-to-host U1 exit latency of 15 µs for t1 (10 µs device exit latency plus 1 µs for each of five hubs), plus 3.1 µs for t2, plus 1.3 µs for t4,

plus some guard band. This assumes the device will not allow its link to enter U2 prior to changing its state to LT-idle. If the device will allow its link to enter U2 when in LT-active, then the total latency the device must be able to tolerate is 125 µs plus the worst case value for U2SEL.

For system implementations where U1SEL is less than its worst case value, the device reports a BELT value larger than 125 µs.

C.4.1.3 Transitioning Between LT-States

When a device transitions between LT-states, the device sends an LTM Transaction Packet (TP) with an updated BELT. The device should send all BELT updates as soon as possible after a change in LT-state.

C.4.1.3.1 Transitioning From LT-idle to LT-active

Devices transition from LT-idle to LT-active when the device determines that a bulk or interrupt data transfer needs to occur. Some examples are given below:

- The host initiates a bulk OUT transfer with the Packets Pending flag asserted to a flash drive device. As a result of receiving this OUT request, the device transitions to LT-active and sends an updated BELT to the host.
- The host initiates a bulk IN transfer with the Packets Pending flag asserted to a hard disk drive device with is spindle currently spun down and the requested data not in a cache on the hard disk drive. As a result of receiving this IN request, the device determines that a bulk data transfer has been initiated by the host. However, the device will service the IN request after its spindle spins up, which may take substantially longer than the last reported BELT. The device may delay transitioning to LT-active. When the device determines that the spindle will complete its spin up within the last reported BELT, the device transitions to LT-active and sends an updated BELT to the host.
- A Network Interface Controller device begins receiving data on its network interface that requires a bulk IN data transfer with the host. As a result of receiving data on its network interface, the device transitions to LT-active, begins to transition its link to U0 (if not already in U0), and sends both an ERDY and an updated BELT to the host.
- A multi-touch Human Interface Device is set up with an interrupt endpoint. When human input is detected the device transitions to LT-active, begins transitioning its link to U0 (if not already in U0), and sends both an ERDY and an updated BELT to the host.

In some cases, a transition from LT-idle to LT-active is not appropriate even though the device needs to transmit to the host. For example:

- The host sends a GetStatus request to a device. Since it is a control transfer and not a bulk or an interrupt transfer, the device remains in the LT-idle state.

When the device transitions from LT-idle to LT-active, the device sends an LTM TP with a BELT of at least tBELTmin.

C.4.1.3.2 Transitioning From LT-active to LT-idle

When a device determines that it is idle, it transitions from LT-active to LT-idle. The method used in this example for device idle detection is based on U2 entry. When the device is in LT-active, the device transitions to LT-idle just prior to when its link will enter U2.

In preparation for a transition from U0 to U2, a device in LT-active should perform the following actions:

1. The device transitions to LT-idle.
2. The device sends an LTM with a BELT value of at least tBELTdefault.
3. The device initiates a transition to U2 from U0.

In preparation for a transition from U1 to U2, a device in LT-active should perform the following actions:

1. The device transitions its link to U0 prior to U2 entry.
2. The device transitions to LT-idle.
3. The device sends an LTM with a BELT value of at least tBELTdefault.
4. The device initiates a transition to U2 from U0.

In the latter case, since the device must transition its link to U0 prior to U2 entry, the device must detect the U2 inactivity timer expiration enough in advance to avoid the possibility that its link partner will have already transitioned from U1 directly to U2. For example, the device may initiate a transition to U0 1 μs before the U2 inactivity timer expires.

C.4.2 Other Considerations

The following are additional considerations associated with device support of LTM:

- The BELT represents a latency tolerance for an entire peripheral device. The BELT value must be aggregated across all endpoints within the device, including all functions within a composite device. The smallest BELT value across all endpoints should be selected. For LTM purposes, isochronous endpoints are ignored when determining the BELT value.
- If LTM is supported by a device, LTM should be disabled prior to placing the device into suspend (refer to the PORT_LINK_STATE feature selector in Chapter 10). Devices send an updated LTM when LTM is enabled, or immediately before LTM is disabled, as defined in Chapter 8. Disabling LTM in this way ensures that a suspended device does not keep the system in a high state of readiness, wasting power.

C.5 SuperSpeed vs. High Speed Power Management Considerations

Some devices may operate well with a High Speed (480 Mbps) interface, but can substantially reduce system power consumption if implemented with a SuperSpeed interface. In addition to device power consumption, system power consumption should be considered when selecting the interface for a new device design.

When a device is actively transferring data, system components are also transferring that data. For some systems, the power consumption of system components is much larger than a USB device's contribution to the system's power consumption.

Under typical circumstances, the faster the data transfer completes, the faster system components can return to a low power state. Transferring data faster can save power, on average, over time. Examples of system components include a Host Controller, a DRAM controller, DRAM components, a microprocessor with a cache that needs to snoop DRAM accesses, etc.

Figure C-9 illustrates a sample device that has an average data transfer rate of 20 MBps when actively in use. The figure shows the system power consumption when the device is operating in SuperSpeed mode and also in High Speed mode.

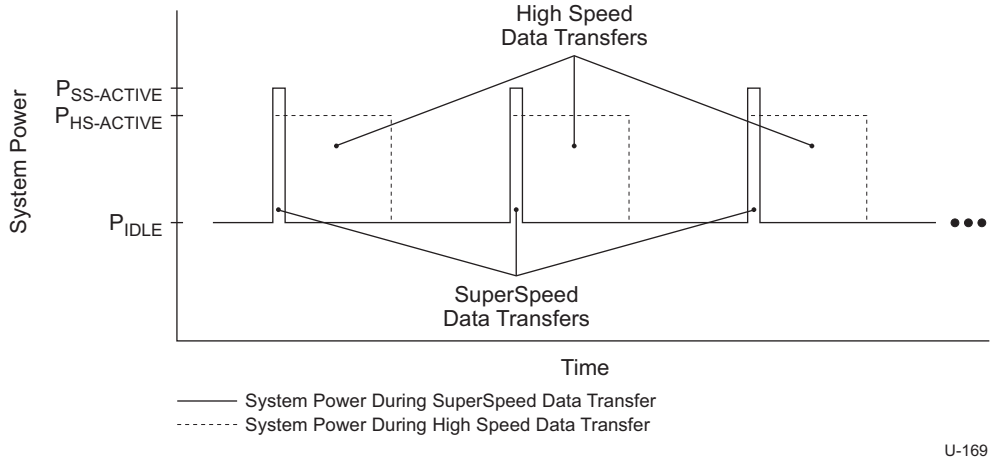


Figure C-9. System Power during SuperSpeed and High Speed Device Data Transfers

When no data transfer is taking place the system power consumption is P_{IDLE}. P_{IDLE} is approximated to be the same in both SuperSpeed and High Speed modes. Link power management considerations are ignored for simplicity of illustration.

When a data transfer is taking place, the system power is $P_{SS\text{-}ACTIVE}$ and $P_{HS\text{-}ACTIVE}$ for SuperSpeed and High Speed modes respectively. The difference between $P_{SS\text{-}ACTIVE}$ and $P_{HS\text{-}ACTIVE}$ is due to the physical layer interface power of the device and its link partner (no hubs present).

Data transfers complete roughly ten times faster in SuperSpeed mode than in High Speed mode. This causes the average system power in High Speed mode to be much larger than the average system power in SuperSpeed mode. The difference in average system power may be as high as 50% during a data transfer. This can have a major impact on the battery life of mobile systems.

D Example Packets

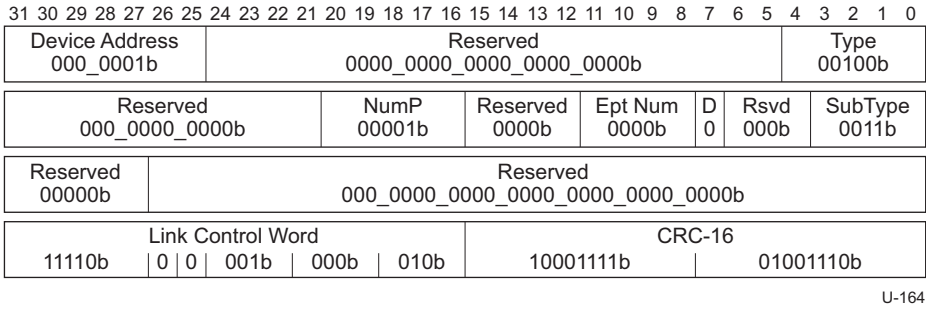


Figure D-1. Sample ERDY Transaction Packet

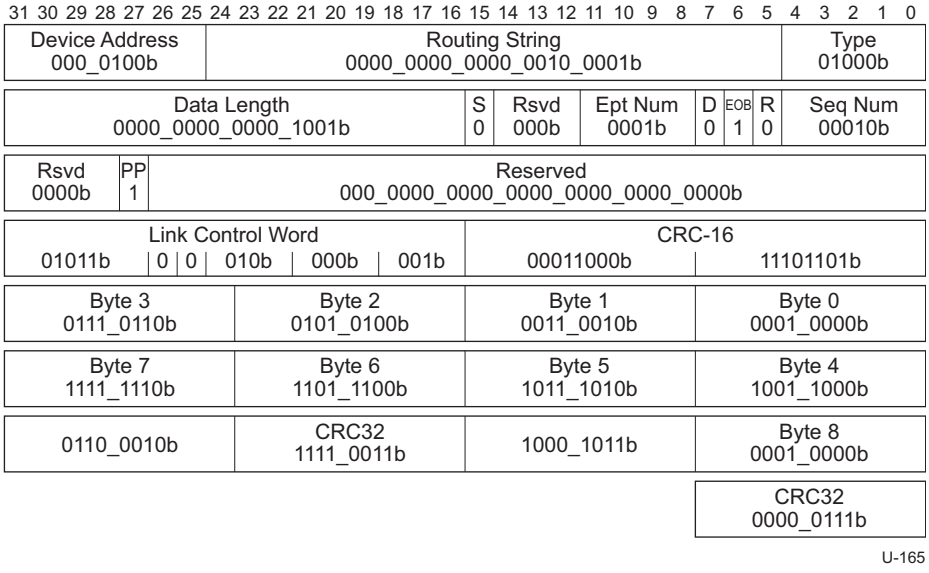


Figure D-2. Sample Data Packet